Debreceni Egyetem

Informatikai kar

Web alapú alkalmazás

Témavezető:
Dr. Rutkovszky Edéné
Egyetemi tanársegéd

Készítette:
Nagy Miklós
Programozó matematikus

Debrecen

2007

Tartalomjegyzék

I. Bevezető

Napjainkban észrevehetjük, hogy olyan gyorsan fejlődik a számítógépes világ. Az informatika területén is egyre újabb dolgok jelennek meg. Nézzük például a programozást! Míg régen a Turbo Pascal, a C nyelvek voltak a csúcson, ma már sajnos eltűnőben vannak. Ennek pedig az az oka, hogy a grafikus operációs rendszerek megjelenésével egyre nagyobb lett az felhasználói igény a grafikus programokra. Az Internet megjelenésével, amikor nagyon sok időt a böngésző használatával töltünk, felvetődött az ötlet, hogy miért nincs olyan program, amit a böngészővel lehet futtatni? Ennek hatására megjelentek különböző programnyelvek. Mint például a Javascript, a Php, stb. Ezeket pedig böngészővel lehet futtatni.

Szakdolgozatom témája egy Web alapú alkalmazás, történetesen egy Web alapú oktatóprogram. Az interneten való tanulásnak nagyon fontos szerepes van mai rohanó világunkban. Időt és energiát spórolhatunk meg vele. Rengeteg féle témában lehetne készíteni oktatóprogramot, de én úgy döntöttem - mivel napjaink elengedhetetlen kellékei az autók - hogy egy KRESZ oktató programot készítek el. Céljaim:

- Kis ízelítő a webes programozásból
- Web-en keresztüli adatbázis-kezelés bemutatása
- A PHP + MySQL + Apache eszközrendszer közelebbi megismertetése

Manapság már szinte lépten-nyomon találkozhatunk adatbázisokkal. Egy komoly program, amely információk ezreit szeretné tárolni, villámgyorsan elérni, és adatokkal dolgozni, elengedhetetlen, hogy adatbázist használjon.

Én a PHP programozási nyelvvel ismerkedtem meg, ami egy nagyon hatásos és kényelmes eszköz az informatika világában. Sikerének több oka is van:

- Könnyű kezelhetőség
- Nagy stabilitás
- Platformfüggetlenség

- És a világháló révén a föld bármely pontjáról elérhetőek a programok

Ha valaki már foglalkozott a PHP programozási nyelvvel, akkor könnyen rájött, hogy milyen egyszerű használni. A legtöbb nyelvhez egy fejlesztői környezetet is fel kell installálni a számítógépünkre. A PHP-nál ezzel szemben egyáltalán nem kell fáradozni semmiféle fejlesztői környezet feltelepítésén. Egyetlen dolog van, aminek lennie kell a gépünkön, az pedig a Web-böngésző.

Szinte teljesen mindegy, hogy Explorert, vagy Mozillát használunk, hogy csak a nagyobbakat említsem, hiszen böngészőtől függetlenül végre lehet hajtani a programjainkat.

Ezt a szakdolgozati témát ezzel a programozási nyelvvel oldottam meg, amit most kicsit részletesebben be is fogok mutatni. Remélem egyre többen fogunk vele foglalkozni mi programozók, ezzel a nagyon kivételes és egyedülálló programozási csodával.

Ha jobban elmélyülünk az adatbázis témában, rájövünk arra, hogy milyen kényelmes, gyors és egyben hihetetlenül hatékony eszközrendszer van a birtokunkban, akkor már olyan programokat írhatunk, melynek csak a képzelet szabhat határokat.

A programot kétféle szemszögből fogjuk vizsgálni:

1. felhasználói részről

2. programozói részről

A felhasználói részben a felhasználó kap egy teljesen átfogó és nagyon érthető magyarázatot, segédletet és útmutatást a programmal kapcsolatban. Olyan leírást próbáltam készíteni, melyet az informatikában kevésbé járatos ember is könnyen meg tudjon érteni.

Programozói szemszögből pedig főleg szakmai vonatkoztatásban kerül a program bemutatásra. Úgy ahogyan azt egy szakmabeli is látja.

A program írása során törekedtem arra, hogy a végeredmény a lehető legjobban felhasználóbarát legyen, hogy kevés számítástechnikai tudás is elég legyen arra, hogy valaki használni tudja ezt az alkalmazást.

Végezetül ejtsünk szót a legfontosabbról. Ezt a programot oktató céllal készítettem. Bárki használhatja, aki úgy gondolja, hogy egy kicsivel is több tudásra tehet szert a program használatával. Mert ilyenkor azt szokták mondani, hogy már akkor is megérte elkészíteni az alkalmazást, ha csak egyetlen egy ember is hasznosíthatta a belőle megszerzett tudást.

Debrecen, 2007. március 9.

II. Fejlesztői eszközrendszer

Php

Mi az a PHP?

A PHP (angol elnevezése "PHP: Hypertext Preprocessor") egy általános célú, nagyon elterjedt scriptnyelv, melyet ma már széles körben használnak. Egyik fontos tulajdonsága, hogy kifejezetten alkalmas - akár HTML-be ágyazott – Web-alkalmazások fejlesztésére is.

Egy példával bemutatva könnyebben meg fogjuk érteni a lényegét:

```
<html>
   <head>
       <title>1. példánk</title>
   </head>
   <body>

       <?php
       echo "Helló, World!";
       ?>

   </body>
</html>
```

Ez a nyelv különbözik sok más script-től melyeket pl.: Perl vagy akár C nyelven írtak. Ahelyett hogy olyan programok írásába kezdünk melyek számos sorból állnak, a PHP-nál elég egy kevés kódot a HTML fájlba ágyazni ahhoz, hogy a kívánt kimenetet eredményezze. A program írása során a HTML fájlban néha szükség lehet arra is, hogy a PHP módot váltogassuk, azaz a fordító lássa, hogy az aktuális utasítás HTML vagy esetleg PHP kód-e. Ekkor speciális kezdő és befejező jelölések közé kell tenni a PHP kódblokkokat.

Ezek a példában látható „<?php” és „?>” jelek. Eközött a két határolójel között lévő kód a böngésző számára egyértelműen PHP kódot jelöl.

A PHP-nak vagy egy nagyon fontos eltérése a kliens oldali nyelvektől, mégpedig az hogy a programkód nem a kliensen, hanem a Web-szerveren fut. A fenti példában látható oldal eredményét megnézve nem tudjuk megállapítani azt, hogy a kimenetet milyen kód állította elő. Lehetőség van arra is, hogy úgy állítsuk be a Web-szervert, hogy a PHP minden HTML fájlt feldolgozzon, PHP blokkokat keresve. Természetesen ezek után már tényleg nem lehet kitalálni, hogy mit is tartalmaz egy-egy program.

Egy nagyon fontos tulajdonság, ami a PHP-t ilyen népszerű programnyelvvé teszi, az az, hogy kezdők számára is végtelenül egyszerű. De persze profi programozók számára is rengeteg lehetőséget nyújt. Ne rémüljünk meg a PHP hosszú szolgáltatatás listáját olvasva, hiszen ez is csak azt bizonyítja, hogy mennyire sokat is nyújt a nyelv. A PHP-ban a még a kezdő is nagy léptekkel tud haladni, és rövid időn belül egyszerűbb script-eket is képes lesz írni. Annak ellenére, hogy a PHP programozásban a szerver oldali fejlesztés kapja a főszerepet, a nyelv ennél sokkal többet tud.

Mit tud a PHP?

A nyelv főleg szerver-oldali scriptek készítésére lett kifejlesztve. Tehát amit egy CGI program el tud végezni, teljesen biztosak lehetünk abban, hogy azt a PHP is képes megvalósítani. Dinamikus tartalom generálása, űrlapon lévő adatok feldolgozása, sütik fogadása vagy küldése – hogy csak egy pár funkciót említsünk. Persze a PHP ennél sokkal többet tud.

A PHP programokat 3 fő területen szokták alkalmazni:

- Szerver-oldali programozás. Ez a PHP hagyományos és fő használati formája. Összesen 3 összetevő szükséges ahhoz, hogy ezt a formát használhassuk: az első és legfontosabb egy PHP értelmező, amit vagy CGI, vagy modul formájában használhatunk. A második egy Web-szerver és a harmadik pedig egy Web-

böngésző. A Web-szervernek megfelelően beállított PHP-vel kell rendelkeznie. A program kimenetét (outputját) a Web-böngészővel lehet megnézni, mégpedig a szerveren keresztül a script elérésével. Ez a 3 komponens, akár egy otthoni desktop gépen is képes elfutni, abban ez esetben, ha nem rendelkezünk internet-kapcsolattal vagy csak ismerkedünk a nyelvvel.

- Parancssori programozás. Lehetőség van arra is, hogy a PHP programjainkat böngésző és szerver nélkül futtassuk. Ha ezt a megoldást választjuk, akkor mindenképpen szükségünk van egy PHP értelmezőre. Ebben az esetben gyakori az, hogy olyan programokat írnak melyek valamilyen ütemező segítségével futtathatóak, vagy akár az is hogy egyszerű szövegfeldolgozó script-eket készítenek.

- Ablakozós alkalmazások írása. Fontos megemlítenünk, hogy a PHP nem a legmegfelelőbb nyelv, ha grafikus alkalmazás írásába kezdünk. Persze ha nagyon jól ismerjük a nyelvet, és szeretnénk használni néhány fejlett PHP szolgáltatást a programjainkban, akkor a PHP-GTK (PHP-Gimp ToolKit) eszközt is használhatjuk programok írására. Ezzel az eszközzel lehetőségünk nyílik arra, hogy platform-független alkalmazásokat készítsünk. A PHP-GTK a PHP egy kiterjesztése, de a hivatalos PHP csomagban nem lehet elérni.

A PHP a legfontosabb operációs rendszereken is képes elfutni, sok Unix változatot, a Linux-ot, HP-UX, Solaris és OpenBSD rendszereket, a Microsoft Windows-t, a Mac OS X rendszert, a RISC OS-t, és másokat is beleértve. Rengeteg Web-szervert is támogat a nyelv beleértve az Apache, Microsoft Internet Information Server, Personal Web Server, Netscape és iPlanet szervereket, az Oreilly Website Pro, Caudium, Xitami, OmniHTTPd, és más szervereket. A szerverek nagy részét a PHP modul szinten támogatja, kisebb részével pedig más CGI szabványt támogató szerverekkel is képes együttműködni CGI feldolgozóként.

Összefoglalva tehát, a PHP használatakor mi magunk választhatunk Web-szervert és operációs rendszert egyaránt. A nyelv abban is szabad kezet ad nekünk, hogy mi válasszunk a függvény-alapú és az objektumorientált programozási technika használata között. A PHP 4-es verziójában sajnos még nem található meg a legtöbb szokásos szolgáltatás, de ezt az 5-ös verzióban a fejlesztők pótolták, és ott már a teljes objektum modell a rendelkezésünkre áll.

A PHP sokkal többet tud annál, hogy HTML kimeneteket állítson elő. Rengeteg dolgot használhatunk programjainkban: képeket, PDF állományokat, Flash mozikat, melyek létrehozására még akár futásidőben is lehetőségünk van. Természetesen nagyon egyszerűen tudunk szöveges kimenetet előállítani, mint például az XHTML, vagy bármilyen XML. A PHP ezeket a típusú fájlokat képes előállítani, elmenteni magán a szerveren ahelyett, hogy közvetlen kiküldje őket. Ez azért jó, mert egy bizonyos szerver-oldali gyorsító-tárat lehet megvalósítani.

A PHP sok jó tulajdonságát említve nem szóltunk a legfontosabbról, ami nem más, mint az adatbázisok széles körű támogatása. Egy weblap készítése - mely adatbázisokat kezel - PHP-vel végtelenül egyszerű. A nyelv a következő adatbázisok mindegyikét támogatja:

Adabas D	InterBase	PostgreSQL
dBase	FrontBase	SQLite
Empress	mSQL	Solid
FilePro (csak olvasásra)	Direct MS-SQL	Sybase
Hyperwave	MySQL	Velocis
IBM DB2	ODBC	Unix dbm
Informix	Oracle (OCI7 és OCI8)	
Ingres	Ovrimos	

A PHP rövid története

A PHP, mint programozási nyelv Rasmus Lerdorf nevéhez fűződik. 1994 őszén alkotta meg, de az első verziók csak annyira voltak jók, hogy figyelemmel tudja kísérni, azt hogy kik látogatnak a weboldalaira. A mások által is használt verzió 1995 elején jelent meg, és a „Personal Home Page Tools" nevet viselte. Ez a verzió mindössze néhány kezdetleges programból állt, és csak néhány speciális makrót értett meg. Ezen kívül még sok olyan eszközt is tartalmazott, melyet gyakran használtak akkoriban a honlapokon. Ilyenek voltak pl.: a vendégkönyv, a számláló, stb. Miután a feldolgozó programot újraírták, 1995 közepén a PHP-t „PHP/FI 2. verzió" névre keresztelték. A FI – ami „Form Interpreter"-t jelent – Rasmus egy másik programjából került a nyelvbe, és arra szolgált, hogy HTML űrlapok információit feldolgozza. A Personal Home Page Tools programját kibővítette az FI támogatással, hozzáadott egy MySQL-t és így jött létre a PHP/FI. Ezt követően a program őrült tempóban fejlődött, és az emberek kódokkal segítették a fejlesztést.

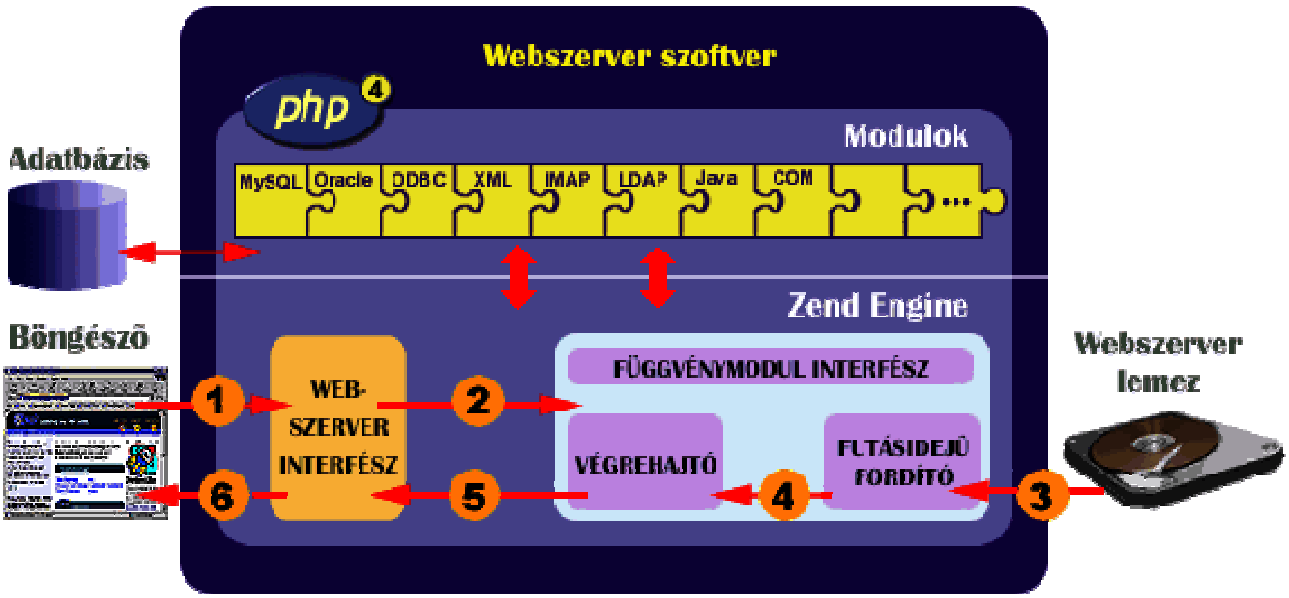
Ha konkrét számokat szeretnénk megadni, akkor azt lehet mondani, hogy 1996 végére a PHP/FI-t már körülbelül 15000 Web-helyen használták világszerte. Az 50000-es határt 1997 közepére érte el. A PHP fejlesztésben 1997 közepe nagy változást hozott, hiszen Rasmus saját kis projektjéből egy jól szervezett csapatmunka lett. Két programozó Zeev Suraski és Andi Gutmans teljesen újraírták a nyelv feldolgozó programját. Így született meg az új PHP 3. Elég sok kódot átvettek a PHP/FI-ből de szintén sok kódot pedig újra kellett írni.

A mai változat a PHP 4-es egy úgynevezett Zend Scripting Engine technológia használatával nagyobb teljesítményt nyújt, ráadásul több kiterjesztést és külső könyvtárat is támogat, valamint minden népszerű Web-szerveren képes elfutni modulként.

Mára a PHP 3 és a PHP 4 programnyelv sok számos üzleti termékkel együtt került forgalomba. Ilyen termék pl.: a Red Hat StrongHold Web-szerver. A NetCraft adatai szerint a PHP-t már több mint 5100000 Web-helyen használják a világ minden táján. Ezzel a teljesítménnyel a PHP megelőzte a Microsoft IIS szerverek számát – Internet Information Server - ami körülbelül 5030000-re tehető.

A PHP működése

Az ábra a PHP oldalak kiszolgálásának folyamatát mutatja be:



Az ábra alapján a PHP script működését a következő elemi lépésekre lehet bontani:

1. A kapcsolat felépítése a Web-szerverhez. Mikor egy felhasználó a weblapra látogat, akkor az történik, hogy a böngészője a Web-szerverhez egy TCP kapcsolatot nyit, azután ezen a kapcsolaton keresztül elküldi a letölteni kívánt oldal jellemzőit és azt is, hogy a választ milyen módon várja.

2. Abban az esetben, ha az oldal egy PHP program, akkor a PHP a Web-szerverhez illeszkedő modulja révén létrehoz egy olyan környezetet mely a PHP script futtatásához szükséges, majd elindítja azt.

3. A konfigurációs fájlban megadott különböző bővítő és illesztő funkciókat ellátó modulokat, és a futtatandó script-eket, az azokban hivatkozott fájlokat egy úgynevezett PHP motor beolvassa a lemezről és az utóbbit a futásidejű fordító felé továbbítja.

4. Az előbb említett futásidejű fordító a script-eket a PHP nyelv szabályainak szintaktikáinak megfelelően próbálja kifejteni és értelmezni. Ha ezt rendben találta, akkor generál egy

köztes kódot, melyet a végrehajtó modul felé továbbít. Ennek a végrehajtó modulnak az a feladata, hogy a már előértelmezett PHP kódból generált programfát végrehajtsa. A PHP script műveleteinek tényleges végrehajtása és a kimenet generálása ebben a fázisban történik. A script - a PHP beépített és a bővítő modulok által biztosított függvényeket használva - információkat írhat a lemezre és olvashat is onnan. Írhat az adatbázisba, de akár még kapcsolatot is nyithat egy másik hálózati számítógép felé, tehát gyakorlatilag bármilyen műveletet végrehajthat. Ennek folyamán generál egy kimenetet mely a böngészőnek visszaküldendő weblap felépítését és a HTML válasz összeállítását jelenti.

5. A PHP motor illesztő modulja továbbítja a Web-szerverhez a script által generált kimenetet. Ezt azért teszi, hogy a kimenetet a böngésző program részére el tudja küldeni.

6. A Web-szerver tehát a böngésző felé továbbítja a kimenetet, ezután a TCP kapcsolatot a beállításoknak megfelelően szétbontja, deinicializálja a PHP futtatókörnyezetét, majd egy olyan állapotba tér vissza, melyből ismét letöltési kérelmeket tud fogadni a kliensektől.

MySQL

Alapok

A MySQL nem más, mint egy több-felhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver. A szoftver kifejlesztője a svéd MySQL AB cég. A fejlesztések 1996-ban kezdődtek mikor a szerzőknek egy olyan SQL szerverre volt szükségük melyek nagy biztonsággal, és meglehetősen gyorsan kezelik az adatbázisokat.

A mai használatban lévő adatbázisszerverek közül talán a MySQL mondtató a legnépszerűbbnek. Nagyon gyakran a PHP-vel együtt használják. Nagy riválisa az ismert PostgreSQL, mely többet nyújt, mint a MySQL, - azaz több szolgáltatása van – de a MySQL-t nagy megbízhatósága és nagy sebessége miatt mégis sokkal gyakrabban használják, még annak ellenére is hogy üzleti felhasználáskor már sok esetben nem ingyenes. Ebből is az látszik, hogy szinte mindenki a megbízhatóságra helyezi a hangsúlyt.

Elérhetősége programnyelvekből

A következő programozási nyelvek mindegyike támogatja a MySQL-t: C, C++, C#, Delphi, Eiffel, Smalltalk, Java, Lisp, Perl, PHP, Python, Rubyés Tcl. Az ODBC-t kezelő nyelvek számára egy úgynevezett MyODBC nevű ODBC interfész teszi hozzáférhetővé az adatbázis-kezelőt.

Adatbázisok Adminisztrációja

Két parancssori eszközt használhatunk adatbázisaink adminisztrációjára, ezek pedig: a MySQL és a MySQLAdmin. Ha grafikus felületen szeretnénk dolgozni, akkor a MySQL Administrator és a MySQL Query Browser programok állnak rendelkezésünkre, melyet a MySQL honlapjáról tudunk letölteni.

A PhpMyAdmin egy nyitott forráskódú és széles körben használt alternatíva. A PhpMyBackUpPro (mely szintén PHP-ban készült) pedig akkor jelent nagyon hasznos eszközt, amikor biztonsági másolatot szeretnénk készíteni adatbázisainkról. Tehát adatbázisok mentésére szolgál.

Stabilitás

A MySQL egy nagyon gyors, több-felhasználós, és többszálas robosztus SQL adatbázis-szerver. 1996 óta a MySQL-lel több mint 40 nagy adatbázist használnak, 10000 táblával. Ezen 10000 táblából mintegy 500-nak több mint 7 millió sora van. Összességében ezek az adatok körülbelül 100 GB tárterületet foglalnak, ami persze a szintén a nagy megbízhatóságot tükrözi.

1999 utolsó hónapjaiban fontos kérdés lett a 2000 kompatibilitás. A Unix idő-funkciók használatával a MySQL tökéletesen 2000 kompatibilis lett. Ez azt jelenti, hogy 1970 és 2069 közötti dátumokat képes kezelni. A YEAR oszloptípus - mely a MySQL 3.22-es változatával került a nyelvbe - már olyan dátumokat is tud kezelni, melyek még az előbb említett tartományon is túlmutatnak. Egész pontosan 1901 és 2155 közötti dátumokat kezel.

Apache

A ma ismert Web-szerverek közül talán az Apache-ot mondhatjuk a legnépszerűbbnek. Egy Web-szerver nem más, mint egy olyan kiszolgáló, mely a helyileg tárolt weblapokat egy HTTP protokollon keresztül teszi elérhetővé. A HTTP Web-szerverekhez való kapcsolódást Web-böngészőkkel szokták megoldani. Egy felmérés szerint a domain-ek 55%-a mögött Apache Web-szerver dolgozik. Ez a számadat tehát jól mutatja az Apache hatalmas népszerűségét. Tovább fokozza ezt a tényt az is, hogy a Microsoft által kiadott Internet Information Server-t a domain-ek csak 22%-a használja. Mivel a Windows operációs rendszer alatt csak béta változatok érhetőek el, ezért viszonylag kevesen használják.

Az Apache Web-szervert honlapjáról a http://www.apache.org címről tudjuk letölteni, de valószínűleg gyorsabban hozzájutunk, ha valamelyik tükörszerverről próbáljuk letölteni.

Most nézzük meg azt, hogy mi is történik valójában a háttérben, amikor a felhasználó a böngészőben egy linkre kattint.

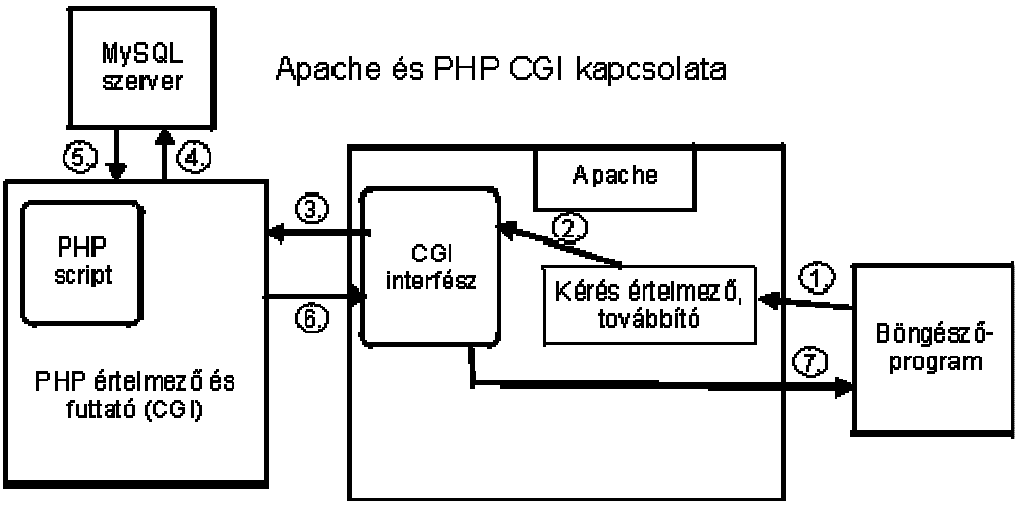
A PHP interpreter futtatása alapvetően 2 módon történhet:

- Vagy úgy, hogy CGI-ként használjuk, mint akár egy PERL programot, vagy bármilyen más standard CGI programot.
- Vagy úgy, hogy Apache modulként futatjuk.

Ennek és annak a módszernek is megvannak a maga előnyei és hátrányai egyaránt. A legnagyobb különbség köztük mégis a PHP script értelmező- és PHP futtató életidejében mutatkozik meg. Ha az Apache-ot CGI-ként használjuk, akkor a scriptek értelmezéséhez mindig egy új interpreter indul el, mely meg is szűnik, miután befejezte a feladatát. Az Apache modul esetében viszont az a helyzet, hogy az interpreter addig marad a memóriában, míg az őt futató Apache kiszolgáló is ott van. Ez sajnos elég hosszú ideig tart, mivel a PHP mindig készen áll és arra vár, hogy akár a felhasználók, akár a Web-szerver feladatot adjon neki. Ez egy kisebb forgalmú oldalon is elég nagy előnnyel jár ahhoz, hogy a PHP-t inkább modulként futtassák. Viszont ennek a megoldásnak van egy nagy hátránya, ami miatt sok

webhely mégis a CGI-s verziót használja, ez pedig az hogy nincs arra lehetőség, hogy más felhasználóként fusson, mint az Apache. Ez akkor jelent problémát, ha például egy szolgáltatónál több fejlesztő használja ugyanazt a szervert, akkor nem oldható meg – modul esetében - hogy a fejlesztők ne tudják elérni egymás állományait.

A következő 2 ábra megmutatja, az Apache működési elvét CGI program esetében



Ez a böngészőtől induló és annál végződő folyamat 7 fő részre osztható:

1. A szervernek a böngésző elküld egy HTTP kérést (az oldal neve legyen például index.php)

2. Ezt a szerver fogadja, értelmezi és eldönti, hogy mit kell kezdeni az oldallal. Ez esetben tudni fogja, hogy tovább kell adnia egy CGI programnak, mivel PHP kiterjesztésről van szó.

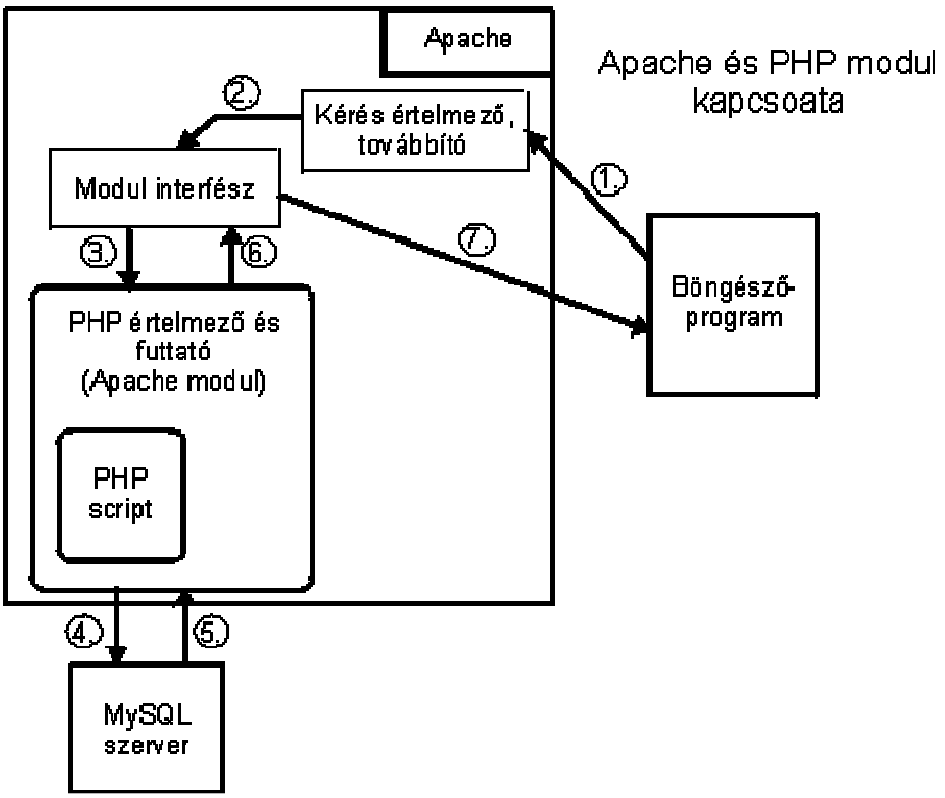
3. A szabványnak megfelelő beállításokkal – környezeti változók, stb. - a CGI interfész elindítja a PHP CGI-s változatát.

4-5. Aközben hogy a PHP más erőforrásokkal is kommunikál, illetve más erőforrásokat is igénybe tud venni, feldolgozza a scriptet.

5. A PHP az előállított adatokat, ami lehet egy HTML oldal, kép vagy akár hang is a standard interfészen keresztül átadja a CGI interfésznek.

6. A CGI interfész pedig ezt továbbítja a böngésző felé.

Most nézzük meg ugyanezt a műveletsort, de most modul esetében:



Az ábráról könnyen leolvasható a két mód közötti alapvető különbség: a hívást a kérésértelmező nem egy CGI interfésznek adja tovább, mint az előző példánál, hanem a memóriában várakozó PHP modulnak, egy modul interfészen keresztül. A különbséget két fontos kulcsszóval lehet leírni: a meghívja és elindítja. Míg a CGI-s változat újraindul minden hívásnál, addig a modul esetén – mivel a modul az Apache része – minden elintézhető egy gyors függvényhívással.

III. Az alkalmazás

Mit kell tudni a programról?

A program egy mindenki számára elérhető alkalmazás, aki rendelkezik internettel. Napjainkban és még régen is nagyon fontos dolog volt egyik helyről eljutni a másikra. Az autók megjelenésével pedig ez valamelyest leegyszerűsödött. De megjelent egy új szabály, amely leírja az autóvezetőknek, hogy miként és hogyan kell szabályosan vezetni. Ez pedig nem más, mint: a KRESZ.
Ennek a szabályrendszernek az elsajátítását szolgálja ez a program is. Ami persze nem elég csak önmagában, hanem a megfelelő tankönyv elsajátítása mellett nyújt kellő tudást a jogosítvány megszerzése érdekében.

Az alkalmazás létrehozása a Web-en

Ezen alkalmazás kapcsán nem beszélhetünk telepítési útmutatóról, hiszen nem egy konkrét programról van szó, melyet egyik számítógépről a másikra ilyen-olyan adathordozón keresztül tovább lehet vinni. Egy általános programnak annyi példánya lehet, amennyit csak akarunk, ennek az alkalmazásnak viszont csak egy. Egy, de ahhoz bárki bárhonnan hozzáférhet.
Bizonyos ingyenes Web-tárhelyek lehetőséget adnak arra, hogy a PHP programjainkat szerverük segítségével futtathassuk le, sőt ennek tetejében még MySQL szolgáltatást is biztosítanak a számunkra. A következő dolgokat kell tenni ahhoz, hogy ezt az alkalmazást mindenki számára elérhetővé tegyük:

1. ***Keresünk egy Web-szervert, mely PHP és MySQL támogatással rendelkezik.***
   Rengeteg Web-szerver létezik, melyek eleget tesznek ezeknek a kritériumoknak.
   Én az ULTRAWEB szerverét választottam. Ez a Web-szerver fogja majd az alkalmazásunkat futtatni.

2. ***Regisztrálunk a Web-szerverre***

Ezt nagyon egyszerűen megtehetjük, hiszen a szerver honlapján – melynek címe: *www.ultraweb.hu* – egy REGISZTRÁCIÓ gombbal csak egy formot kell kitöltenünk. Ez egy egyszerű kérdőívhez hasonlítható: a felhasználói nevet, személyes adatokat és a honlap adatait fogja majd kérni a form. Ezen adatok ismertében fog minket az ULTRAWEB regisztrálni. Ha a böngészőnk URL-jébe a megadott felhasználói név után hozzáillesztjük a „.uw.hu" sztringet akkor már hivatkozhatunk is a weblapunkra. (Ami persze még teljesen üres). Formálisan: *www.felhasznaloi_nev.uw.hu*
Regisztrációnkhoz kapunk: 200 MB ingyenes tárhelyet, 20 MB MySQL tárhelyet melyen az adatbázis tárolódik, SQL PHP futtatási lehetőséget, FTP hozzáférést,stb.

3. ***Feltöltjük az adatbázist az erre megfelelő program segítségével***

Miután bejelentkeztünk a szerverre az ULTRAWEB honlapján az előzőleg megadott azonosítónkkal, annyi dolgunk van, hogy létrehozzuk és feltöltsük az adatbázist. Erre egy úgynevezett PHPMYADMIN nevű script áll rendelkezésünkre. Ebben meg kell adnunk a létrehozandó adatbázis nevét, és a használt karakterkészletet. Miután ez kész, az adatbázis nevére kattintva megtekinthetjük annak tartalmát. Látni fogjuk, hogy még teljesen üres. Táblákat kell létrehoznunk. Ehhez 2 dolgot kell megadni a scriptnek: 1. a tábla nevét, 2. a táblában lévő mezők számát. Ezután a script az összes mező típusára és hosszára rákérdez, melyeket egyenként meg kell adnunk, sőt egy mezőt ki kell választanunk elsődleges kulcsnak. Ha elkészült, akkor már tényleg jöhet a tábla feltöltése rekordokkal. Megjelenik egy kis szövegmező a következő felirattal: „SQL parancs(ok) futtatása a(z) nagymiki2 adatbázison". Ide be lehet írni az utasításokat, melyekkel fel akarjuk tölteni a táblát. Az elég nagy problémát jelenthet, ha sok rekord van mert több száz, vagy esetleg több ezer SQL utasítás végrehajtása külön-külön nem könnyű-, és persze nem rövid feladat. Erre is van megoldás: egy szöveges állományban helyezzük el soronként a végrehajtani kívánt utasításainkat. Majd ezt a szövegfájlt kell megadni a programnak, amely sorokként végre is hajtja az egészet. Na ugye máris mennyivel egyszerűbb a dolog? Ha ez lefutott, akkor a tábla fel van töltve.

4. ***Feltöltjük a PHP fájlokat***

Mivel a Web-szerver FTP szolgáltatást is biztosít, ezért nagyon kényelmesen fel tudunk rá tölteni bármit, amit akarunk. Elindítjuk a kedvenc fájlkezelő programunkat, és azon keresztül fogunk kapcsolódni az ULTRAWEB szerveréhez.
3 fontos dolgot kell megadni a kapcsolódáshoz: a kapcsolat nevét (bármilyen karaktersorozat), a kiszolgáló nevét (kötelezően: *ftp.uw.hu*), és a felhasználói nevet (melyet még a regisztrációnál adtunk meg). Innentől kezdve már csatlakozhatunk is a Web-szerverhez. Az utolsó momentum, hogy átmásoljuk rá a PHP programokat. NAGYON FONTOS: A szerveren lennie kell egy index.html nevű fájlnak is, mert a szerver ezt fogja keresni, mikor a URL-be beírjuk a címet:
www.felhasznaloi_nev.uw.hu. Ha nem adunk meg ilyen fájlt, akkor hibaüzenetet kapunk.

Az alkalmazás elérhetősége

Nagyon egyszerű a felhasználó dolga, mert nem kell mást tenni, mint feltelepíteni egy számára kényelmes, megszokott böngészőprogramot és már készen is van. Egyetlen dolga van ezután a felhasználónak: beírni azt a címet melyről ez a Web-alkalmazás elérhető. Ez pedig nem más, mint a:

www.nagymiki2.uw.hu

Ekkor megjelenik az a Web-felület, amelyen a program fog futni. Innentől már minden készen áll és a felhasználó már használhatja is a programot.

A program felépítése

A. Felhasználói szemmel

Mikor a böngészőbe beírjuk a következő URL címet: www.nagymiki2.uw.hu akkor a Web-szerver elindítja az alkalmazást.

A következő képernyőt fogjuk látni:

Teljesen egyszerű a kezelés, mert a programban rengeteg helyen található magyarázat, mely nagymértékben segíti akár még a kezdő felhasználóknak is a program használatát.
Nem sok választási lehetőség van, összesen 2 db link. Ez a két link a képernyő felső harmadában helyezkedik el, amit két darab kék forgó nyíl is jelez. A linkekhez a következő 2 programegység tartozik:

a) KRESZ- teszt
b) KRESZ-vizsga

A programot tehát 2 részre bontottam, azért hogy mindenki annak megfelelően válasszon feladatot, hogy éppen mit szeretne gyakorolni. Ha csak gyakorolni szeretne, akkor válassza a KRESZ-tesztet. Ha viszont már elege van a felhasználónak a gyakorlásból és szeretné már végre kipróbálni mit sajátított el, akkor pedig a KRESZ-vizsga linkre kell kattintani. Most az elkövetkezőkben meg fogjuk nézni kicsit közelebbről ezt a két részt.

KRESZ-teszt

Alapok

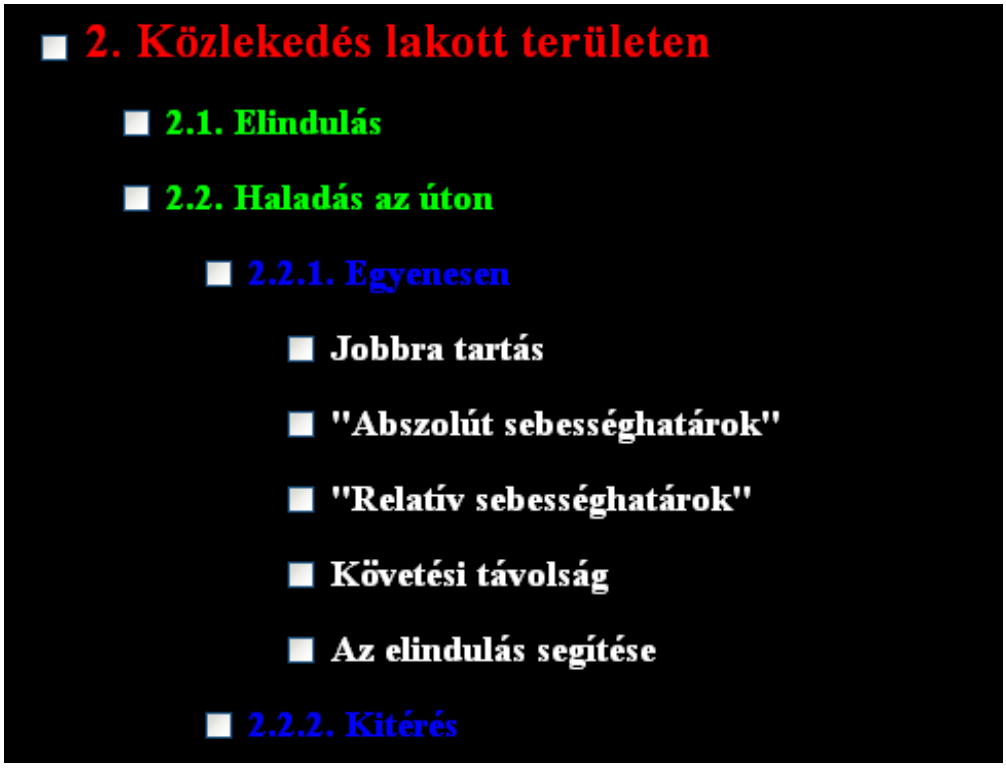
A KRESZ-teszt nevű alkalmazás tulajdonképpen hasonlít a KRESZ tankönyv tesztes kötetéhez. Itt is ugyanúgy tesztelhetjük tudásunkat, mint a könyvben.

Miután kiválasztottuk, hogy a KRESZ-teszt fejezettel akarunk dolgozni, akkor a teszt-könyvhöz hasonlóan elénk tárul egy teljes lista, ami a teszt-könyv összes témakörét tartalmazza.

Minden témakör előtt egy jelölőnégyzetet találunk. Ezzel kiválaszthatjuk, hogy milyen témakörrel szeretnénk foglalkozni. 4 részre lehet osztani, a teljes fejezetlistát:

1. Kérdéstípusok
2. KRESZ
3. Vezetéselmélet
4. Műszaki

Ez a 4 típus segít abban, hogy a számunkra megfelelő kérdésekre szűkítsük a rengeteg kérdést. A kérdések hierarchikus formában jelennek meg a képernyőn.



A főtémák piros színnel, ezek altémái pedig zöld, azután kék végül pedig fehér színben jelennek meg. Számos kombinációban alkothatunk kérdéscsoportokat. Úgy működnek ezek a jelölőnégyzetek, mint valami szűrők.

Példa: a követési távolság témát szeretnénk gyakorolni, de semmi más egyebet. Ekkor a „követési távolság” előtti jelölőnégyzetet bejelölve tudjuk elkezdeni a gyakorlást. De ez a követési távolság című fejezet az „egyenesen” című fő témának az egyik alpontja. Szóval, ha az „Egyenesen” téma előtt jelölőnégyzetbe teszünk pipát, akkor természetesen az abba a témába tartozó összes témát fogja tartalmazni a kérdés-összeállítás. Azaz a „Jobbratartás”-tól az „Elindulás segítése” témáig mind az 5 db-ot. De ha nem mind az 5 témát akarjuk, akkor értelemszerűen többet is be lehet jelölni, az altémákból.

Előfordulhat az az eset is, amikor a felhasználó csak és kizárólag a táblákat szeretné gyakorolni.

Ilyenkor a kérdéstípusok csoportban csak annyi dolgunk van, hogy a „csak táblás” jelölőnégyzetet bejelöljük. Ilyenkor az összes témából csak az olyan kérdéseket szűrtük ki melyekben tábla van.

Viszont ha nekünk csak az olyan kérdések kellenek, amelyek csak autópályás kérdések és azon belül is csak táblás akkor azt is nagyon könnyen megtehetjük a mindkét jelölőnégyzetet bejelölve.

Felmerül tehát a kérdés, hogy mi van akkor, ha néhány vicces kedvű felhasználó bejelöli a táblás és ábra nélküli kérdéseket egyszerre? Természetesen erre az esetre is fel van készítve a program. Azaz logikailag egymásnak ellentmondó témákat nem lehet kiválasztani.

Hibakezelés

1. Ábra nélküli és (táblás / szituációs / forgalmi helyzetes / egyéb ábrás) kérdéscsoport egyszerre való kijelölése esetén, a következő hibaüzenet jelenik meg:

2. Az az eset is előfordul, ha a felhasználó nem választott ki semmilyen témát, ekkor a következő szöveg jelenik meg:

3. Sajnos van, mikor a program nem talált kérdést az általunk kiválasztott kérdéscsoportban. Ha a matematika nyelvére szeretnénk ezt lefordítani, akkor olyan szűrőt adtunk meg, amely által létrehozott eredményhalmaz számossága zérus.
   Ekkor ezt a hibát kapjuk:

Tehát a program minden eshetőségre fel van készítve. Bármilyen kombinációban szerkeszthetünk magunknak kérdéseket. Persze a logikailag hibás összeállításokért amint az előbb is láttuk a program hibaüzenettel tér vissza és visszalinkeli a felhasználót a főmenühöz. Ott ismét választhat, addig ameddig logikailag jó nem lesz a választott témakör.

Ha mindent rendben talál a program, akkor átengedi a felhasználót egy következő felületre, ahol már a konkrét kérdések vannak. Itt a már lehetőségünk van belevetni magunkat a kérdésekbe, ahol feladatok tömkelege várja a felhasználótól a helyes megoldást.

A dolgozó felület

Ha kiválasztottuk a megfelelő témakört, akkor egy új felület tárul elénk, ez pedig már a tényleges „gyakorló felület”. Ez különböző részekből áll. De nézzük meg, hogyan is néz ez ki:

Teljesen már színnel, más felülettel kell megbarátkoznunk, ugyanis ezen a felületen fogunk a legtöbb időt eltölteni. Itt kell megválaszolni a számítógép által feltett kérdéseket.

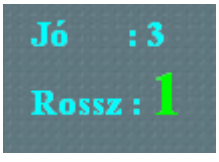
A felület berendezkedése

3 fő részt különíthetünk el, ezek:

1. Információs rész
2. Teszt rész
3. Jóváhagyás/továbblépés illetve befejezés rész

Az információs rész tartalma, feladata nem más, mint az hogy információkat közöljön a felhasználó számára. Ez pedig a témakör megnevezése, a jó illetve a rossz válaszoknak a száma, valamint a figyelmességre való felhívás.
Középen van sárga színnel a témakör. Ez most konkrétan ebben az esetben az „Abszolút sebességhatárok". Ha elérünk egy másik témát, akkor ez a téma automatikusan egy másikra átvált.
A jó és rossz válaszok pedig azt mutatják, hogy ez idáig mennyi jó illetve rossz kérdést válaszoltunk meg. Kezdetben 0:0-ról indul a számláló. Mind a kettő kicsi betűmérettel és ciánkék színnel jelenik meg:

Ezek után már kettő közül az egyik kicsit nagyobb betűmérettel és más színnel jelenik meg:



Ez is úgymond egyfajta információhordozó, és a jelentése pedig a következő: Eddig 4 feladatot oldottunk meg ebből 3 jó és 1 pedig rossz. Az pedig, hogy a rossz válaszoknál lévő 1-es számjegy nagy betűmérettel és zölddel jelenik meg, az azt jelenti, hogy a legutolsó válaszra milyen minősítést kaptunk. Ebből a konkrét esetből az látszik, hogy az utolsó válaszunk sajnos nem sikerült, rossz választ adtunk rá.

A felhívás pedig csak annyi, hogy figyelmesen olvassuk el a feladatot, és csak úgy jelöljünk, és hogy csak 1 helyes válasz létezik.

Ezt a részt természetesen nem módosíthatja a felhasználó. Nem tudja átírni, sem pedig bármilyen módon manipulálni, mert a memóriában tárolódik.

Beszéljünk kicsit a Teszt részről. Ez a következőkből áll:

- Aktuális kérdés/Összes kérdés: Ez mutatja meg azt, hogy a számítógép által feltett kérdések közül hányadiknál járunk, és az általunk összeállított feladatcsokor hány kérdést tartalmaz (Pl.: 17/54 jelentése, hogy a 17. kérdésnél járunk az 54-ből)
- Ezt követi maga a kérdés, szöveges formában.
- Ha a feladathoz kép is társul, akkor azt is megjeleníti a program.
- A kép alatt pedig állnak a lehetséges válaszok. Mindegyik előtt vagy egy radiobutton nevezetű elem, aminek olyan tulajdonsága van, hogy egy választ lehet vele bejelölni.

A kérdés megválaszolása nem időre megy, időt hagyva ezzel a teljesen kezdők számára. A cél, hogy a program minél jobban felhasználóbarát és könnyen kezelhető legyen. Ne kedvetlenítsen el senkit. Mert a legfontosabb cél az, hogy mindent elkövessünk a felhasználó eredményessége érdekében, és ha már az elején a kedvét szegi a nehezen kezelhető program, akkor nemhogy a programot nem fogja már soha többet használni, de legrosszabb esetben még az egész tanulástól, a KRESZ-től is teljesen elmegy a kedve. Szóval mivel van idő bőven, a felhasználó kedvére gondolkodhat, de azután meg kell adni a választ, mert addig nem tudja továbbfolytatni a tesztet.

A jóváhagyás/befejezés rész: Ebben az egységben a program felajánl 3 lehetőséget, ami közül nekünk kell döntenünk:

- ha az OK gombra kattintunk, akkor egyértelműen jelezzük a program számára, hogy tovább akarjuk folytatni a tesztet. Kíváncsiak vagyunk a válasz értékelésére.

- A „Vissza a főmenühöz” link, mint a neve is mutatja, visszajuttat minket a főmenühöz.
- ha viszont „A gyakorlás befejezése” linkre kattintunk, akkor a kérdés javítása nélkül visszakerülünk oda, ahol kérdéseket tudunk válogatni. Azaz az eddigi kérdéscsokrot elvetjük. Vigyázat! Ez a gomb NEM a főmenühöz juttat vissza minket.

A javítás

Abban az esetben, ha rákattintottunk az OK gombra, akkor a következő képernyő majdnem teljesen ugyanúgy fog kinézni, mint amelyikben kiválasztottuk a választ:

A különbség csak annyi, hogy eltűntek a radiobutton-ok, és megjelent egy zöld színű pipa, és egy piros színű X jel, HELYTELEN felirattal. Továbbá a rossz válaszok száma a fenti információs mezőben 1-gyel megnőtt. Na mit is jelenthet ez? Azt, hogy sajnos rossz választ adtunk a feladatra, amit a számítógép ki is javított. A mi általunk bejelölt válasz mellett most piros X jelent meg, a jó válasz mellett pedig zöld pipa.

Van azonban a másik eset is, amikor jó a válaszunk akkor a következőt látjuk:

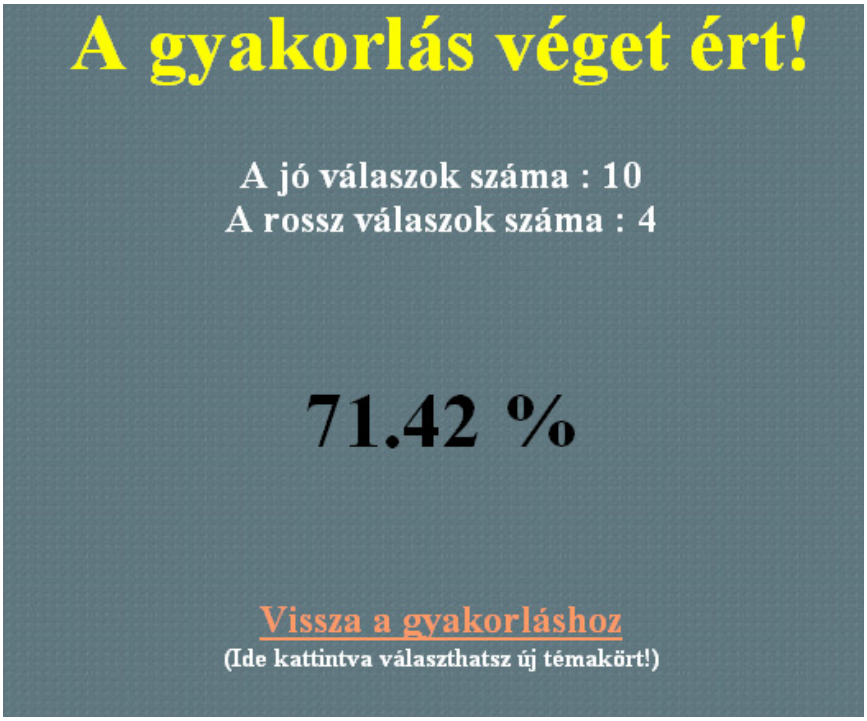
Ebben az esetben pedig csak egyetlen zöld pipát látunk, mellette a helyes felirattal. A jó válaszok száma 1-gyel megnőtt.

Nagyon fontos az is megemlíteni, hogy mi van akkor, ha a felhasználó semelyik választ sem választja? Mi van, ha csak egyszerűen az OK gombra klikkel? A válasz egyszerű: a program továbblépteti a felhasználót, de a rossz válaszok számát 1-gyel növeli.

Megjelent a „Tovább” gomb, amire klikkelve tovább mehetünk a következő, ez esetben a második feladatra.
A „Gyakorlás befejezése” linkkel pedig visszajutunk a teszt részhez, ahol egy másik témát választhatunk.
Ha végképp be akarjuk fejezni a gyakorló részt, akkor „Vissza a főmenühöz” a főmenühöz jutunk.

Összesítés

Ha tovább megyünk, akkor előbb utóbb eljutunk a feladatsor végére. Itt lehet látni, hogy összesen 14 darab feladat van ebben a gyakorlásban. Csináljuk végig! A legutolsó kérdés javítása után ez az ablak jelenik meg:



Ez egy összesítő tábla. Kiírja az eredményt, hogy hány feladatra adtunk jó, illetve hány feladatra adtunk rossz választ. A biztonság kedvéért kiírja százalékos formában is, mert így sokkal jobban szembetűnik, hogy körülbelül hány százalékig tudjuk a kiválasztott kérdéskört. A „Vissza a gyakorláshoz” gombbal jutunk vissza a gyakorló képernyőhöz.

KRESZ-vizsga

Alapok

A KRESZ vizsgát akkor kellene választaniuk a felhasználóknak amikor, már úgy érzik eleget gyakoroltak a KRESZ-teszt résszel. Itt már nincs közvetlen javítás válaszadás után. A számítógép kiválaszt 55 kérdést különféle témákból. Mikor kész azt tárja a felhasználó elé.

Mikor rákattint a felhasználó a „KRESZ-vizsga” linkre, akkor elindul a program vizsga része. De előbb még a következő képernyővel találjuk szembe magunkat:

A számítógép figyelmeztet minket a legfontosabb dolgokra, amit nem szabad elfelejtenünk: Leírja, hogy hány kérdésből áll a vizsga, mely kérdések érnek 3 pontot és hogy az időnk sem végtelen.

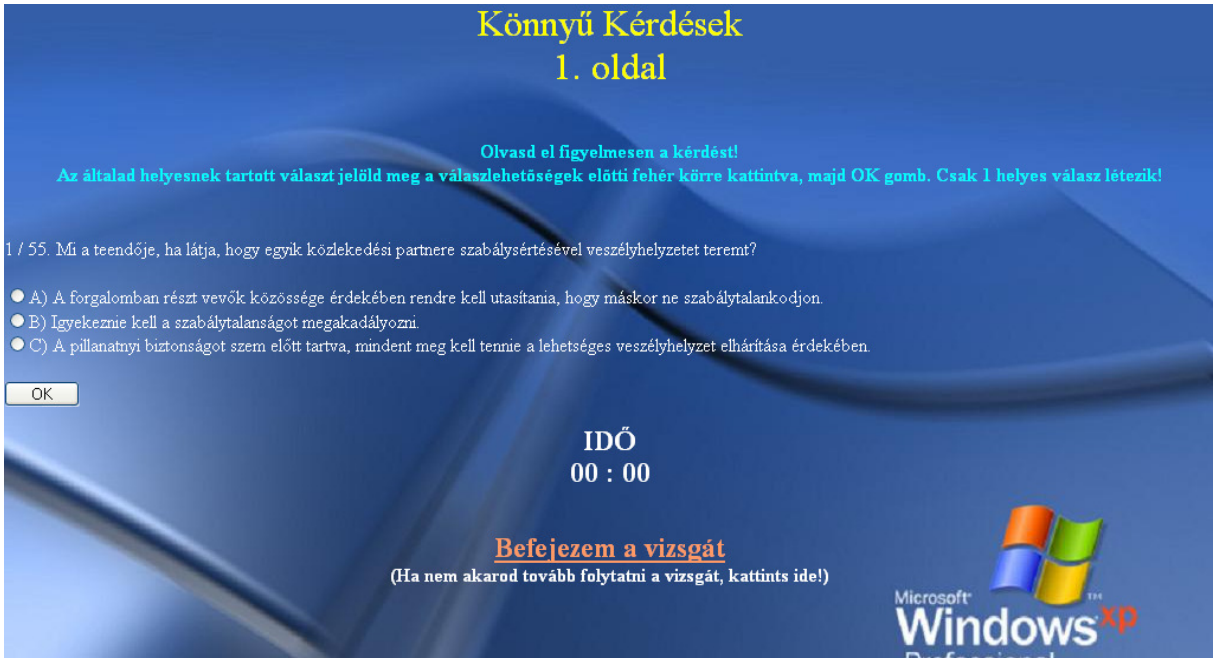
Mikor elolvastuk a szöveget, akkor a linkre klikkelve a számítógép már fel is teszi az 55 kérdésből álló tesztsor 1. kérdését.

A KRESZ-lap szerkezete

első oldal

Az első 10 kérdés a könnyű részből való. Ez azt jelenti, hogy minden vizsga kezdetén a számítógép a főbb alapfogalmakból állít össze egy 10 kérdéses tesztet. Vigyázzunk, mert ha rontunk, akkor egy rossz válaszra 1 hibapontot kapunk.

Ennek teljesen más a felülete, mint a KRESZ-teszt gyakorló résznek.
Külsőleg is megváltozott és működésben is különbségek vannak.



Ezt a munkaképernyőt fogjuk látni 55 kérdésen keresztül, annyi különbséggel, hogy a fenti sárga téma mindig változik. Ezek a „könnyű” típusú kérdések.

Emlékszünk még ugye, amikor a vizsgát nem számítógépen, hanem tesztlapon kellett megoldani. Na az itt megjelenő „1. oldal” a tesztlap oldalszámát próbálja a felhasználó elé vetíteni.

Miután eleget néztük és megbarátkoztunk a felülettel, nincs más dolgunk, mint belevetni magunkat a tesztkérdések megoldásába.

második oldal

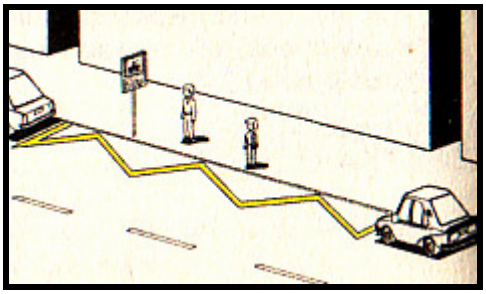
Ezen az oldalon ismét 10 kérdés található. Ezek az úgynevezett táblás kérdések. Itt mindenféle táblával találkozhatunk, ami a közúton, a valós életben is megtalálható. Persze nem csak a táblák nevét kell tudni, hanem minden nemű fontos tudnivalót, ami a táblával kapcsolatos. Pl.: ha előzni tilos táblát mutat a feladat, akkor nem elég azt tudni, hogy mi a tábla neve, hanem olykor olyat is kérdezhet a feladat, hogy mit szabad-, illetve mint nem szabad előzni.

A táblás kérdéseknél egy rossz választ 1 hibapontnak számít a program. Kihagyott kérdés szintén 1 hibapont.

harmadik oldal

Szituációs kérdések. Nehezen határozható meg pusztán szavakkal ez a témakör. Mert nagyon hasonlít a forgalmi helyzetes részre, ami a következő témakör lesz. De egy példával illusztrálva elmagyarázható.

Itt látható egy kép:



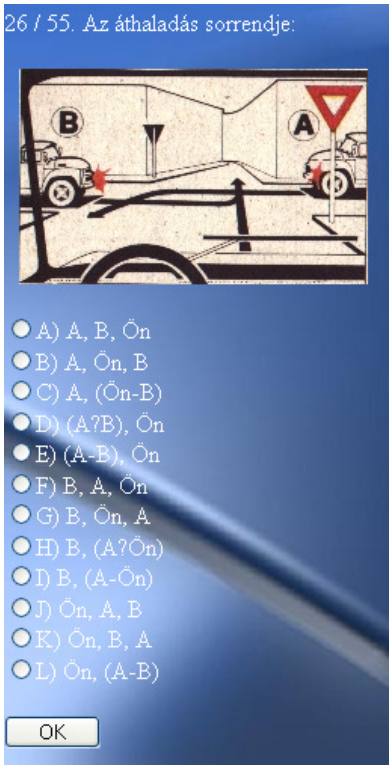
A kérdés pedig az, hogy : „Megállhatunk-e itt utasunk ki- és beszállásának idejére? Igen/Nem.” Láthatjuk, hogy ez tényleg egy forgalmi „szituáció”. Nincs más külső forgalom, csak mi. Döntenünk kell, hogy az adott helyzetben (szituációban) mi a helyes viselkedés.

Nagyon fontos!

A szituációs kérdésekből 5-öt kell megoldani. Viszont ezeknél a kérdéseknél egy rossz választ 3 hibapontnak számít a program. Kihagyott kérdés szintén 3 hibapont.

negyedik oldal

Forgalmi helyzetes kérdések. Itt az esetek 90%-ában egy útkereszteződés, illetve útbecsatlakozás (jobbról illetve balról) látható. Nézzük a következő képet:



Láthatjuk, hogy ebben az esetben egy útkereszteződést kaptunk feladatként. És a kérdés, mint a legtöbb esetben: Az áthaladás sorrendje.
Amint látjuk, nem csak 3 válaszlehetőség van, hanem egy tucat. Lehetne kérdezni, hogy miért van erre szükség? Azért mert ezek olyan fontos és olykor veszélyes helyzetek, hogy a vezetőnek száz megoldás közül is tudni kell, hogy mit kell cselekedni. Azt lehet mondani, hogy 100%-ig biztosnak kell lennie magában.

Meglehet, hogy van a felhasználó, talál egy-két érthetetlen karaktert a válaszokban. Ezek pedig a: „-„ és „?” jelek. Ezek magyarázata a következő:

„-„ : Nézzük a C választ: A, (Ön-B). Ez a következőt jelenti. Elsőként elmegy az A jelű jármű, majd Ön és a B jelű egyszerre indulhatnak el. Tehát úgy kanyarodnak, hogy egymást nem akadályozzák.

„?” : Itt pedig nézzük a D választlehetőséget: (A?B), Ön. Ez azt jelenti, hogy a közlekedés A illetve B jelű résztvevőjének egymás közt kell megegyezni, hogy ki

induljon elsőként, azaz hogy kié legyen az elsőbbség. Ezt akár kézzel való jelzéssel, intéssel is megtehetik. Ezután indulhat el Ön a járművével.

A forgalmi helyzetes kérdésekből 5 darabot tesz fel a számítógép. Ennél a kérdéstípusnál is egy rossz választ 3 hibapontnak számít a program. Kihagyott kérdés szintén 3 hibapont.

ötödik oldal

Vegyes kérdések: Ezek a kérdések a teszt-könyv bármely részéről lehetnek. Persze csak azok, amik nem könnyű-, táblás-, szituációs-, vagy forgalmi helyzetes kérdések! Főként szöveges, kép nélküli kérdések.

A vegyes kérdésekből 10 darabot kapunk. Egy rossz választ 1 hibapontnak számít a program. Kihagyott kérdés szintén 1 hibapont.

hatodik oldal

Járművezetés elmélete. Elméleti, helyenként fizikai feladatokat kapunk a számítógéptől. Pl.: 2x-es sebességről fékezve hányszorosára nő a fékút. Ez inkább fizikai jellegű kérdés. 10-et kell megoldanunk. Itt 1 hibapont jár a rossz válaszokért. A kihagyott kérdések is 1 hibapontnak számítanak. Összesen 188 darab kérdésből választ a program véletlenszerűen.

hetedik oldal

Ez az utolsó téma, amivel a vizsgázónak meg kell birkóznia. Ez a műszaki alapismeretek kérdéscsoport. A program 53 kérdésből választ véletlenszerűen 5-öt. Minden rossz válasz és kihagyott kérdés is 1 hibapontnak felel meg.

A vizsga vége

Nos, nagy nehezen elértünk az 55 kérdés végére. Most derül ki, hogy elég rátermett-e a vizsgázó! Mielőtt a számítógép kijavítaná a feladatokat, azelőtt a következő képernyőt fogjuk látni:

Ez azt jelenti, hogy a számítógép nyugtázza a felhasználónak, hogy kész van. Sikerült végigcsinálnia a feladatsort. A másodperc törtrésze alatt ki is javítja a feladatokat, amit meg is mutat a vizsgázónak. Az „Itt megnézheted” linkre klikkelve már kijavítva láthatjuk a feladatokat.

A kijavított tesztlap

Így néz ki tehát a kijavított vizsga egy része. A program megmutatja, hogy mit is rontottunk el. Mit csináltunk rosszul.

- Nézzük az első feladatot: Az „A” választ jelöltük meg, ami helytelen, kaptunk rá 1 hibapontot és a számítógép meg is mutatja, hogy mi lett volna a jó válasz. (C). Ezzel a feladattal a vizsgaidőből 11 másodpercet töltöttünk el.

- A második feladatban semmit sem jelöltünk be. Kiírja, hogy „Kihagyva” és ismét 1 hibapont. Ezzel 16 másodperc ment el.

- A harmadik feladatot sikeresen megoldottuk. Egy zöld színű pipa jelenik meg a jó válasz mellett és a „Helyes” felirat. Itt nem írja ki, hogy hány jó pontot kapunk. A feladat csak a hibapontokat számolja össze. 13 másodperc alatt adtunk helyes választ a feladatra.

Az értékelés

Miután végignéztük, értelmeztük és reméljük megtanultuk a hibáinkat, hogy legközelebb ne ejtsünk hasonló hibát, akkor a lap alján látunk egy linket, a következő felirattal: „A vizsgabizottság döntése”.
Erre klikkelve már ténylegesen megtudjuk, hogy vajon sikeresen átmentünk-e a KRESZ vizsgán vagy sem. Nos lássuk:

Sajnos ez azt mutatja, hogy nem sikerült a vizsga. Próbálkozzunk újból!
Ezen a KRESZ vizsgán, és élesben is a következő ponthatárok vannak érvényben:

- Sikeres a vizsga, amennyiben 10 vagy annál kevesebb hibapontot szereztünk.
- Sikertelen, ha 10-nél több hibánk van.

Vigyázzunk! Ez nem azt jelenti, hogy akár 10 feladatot is elronthatunk, és mégis átmegyünk a vizsgán. Mert gondoljunk csak bele, 4 db 3 pontos feladatot elrontunk, és már meg is buktunk.

Persze arra törekszik az ember, hogy minél jobban sikerüljön a vizsga. A másik véglet pedig így néz ki:

Ekkor már lehet örülni a sikernek, hiszen arról tájékoztat ez esetben a program, hogy sikerült a vizsga. Sikerült a 75 pontos, 55 feladatból álló tesztből kevesebb, mint 11 pontot hibázni. Mert ugyebár még 10 hibaponttal sikeresnek mondható a vizsgaeredmény. Ezután visszakerülünk a főmenühöz.

Az idő mérése

„Az idő pénz” – tartja a mondás. Mint oly sok vizsga, ez a KRESZ-vizsga is időhöz kötött. Nagyon fontos észben tartanunk, hogy egy bizonyos időhatáron belül kell megoldanunk a számunkra elkészített feladatsort.
55 kérdésből áll a vizsga, minderre összesen 55 perc áll rendelkezésre. Annyi könnyítéssel, hogy az idővel mi gazdálkodhatunk. Tehát ha valaki úgy gondolja, hogy minden kérdésre átlagban 1 percet szán, akkor is meg tudja oldani a feladatokat. Ha viszont valaki úgy szeretné, hogy a könnyebb kérdésekre gyorsabb választ ad, - ezzel is spórolva az idővel – akkor nyilvánvalóan a nehezebb kérdésekre több ideje marad.
Azon okból kifolyólag, mivel az időnket mi osztjuk be, a program a kihagyott kérdéseket nem teszi fel újból a vizsga végén.

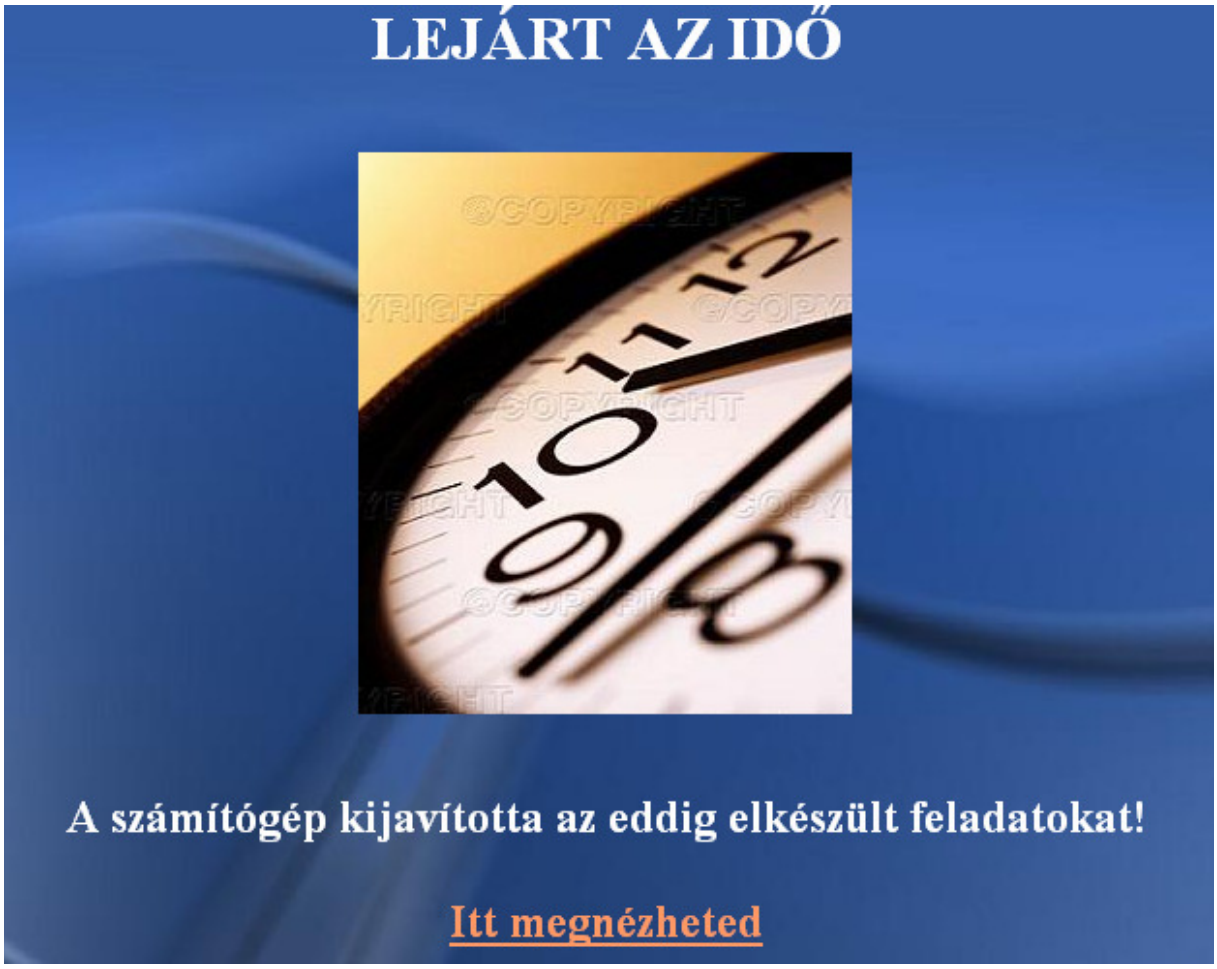
Amikor elkezdődik a vizsga az időmérő alapállásból, azaz teljesen nulláról indul. Ilyenkor a képernyő alján a következő képet fogjuk látni:

Ez a „perc:másodperc” formátum mutatja az eltelt időt. Természetesen az első a perc a második a másodperc. Fontos megjegyezni azt is, hogy nem másodperc frissítéssel jelenik meg az idő, hanem minden lap frissítésnél. Azaz akkor fogjuk látni, hogy mennyi idő telt el az első kérdéssel, mikor a következő kérdésre térünk.
Tehát az első kérdésnél a „00:00” időt láttunk, és a számláló mindaddig ennyit fog mutatni, míg a második kérdésre nem térünk. Ahol már a következőt látjuk:

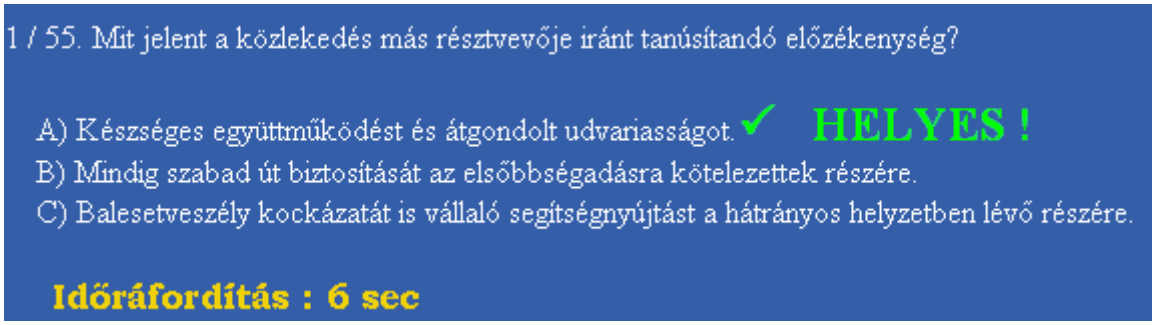
Ez tehát annyit jelent, hogy az aktuális időpont 23 másodperc, tehát az első kérdést 23 másodperc alatt válaszoltuk meg.

Ezzel a dologgal nincs semmi baj, ha időn belül vagyunk. Azaz ha mind az 55 kérdést 55 perc alatt sikerül megoldani. A számítógép kijavítja a feladatokat és értékel. Minden a megadott terv szerint megy.

Nézzük viszont azt az esetet, amikor ez nem valósul meg, ha nem sikerül 55 percen belül megoldani a feladatsort. Mikor a következő kérdésre lépnénk, holott már lejárt az idő akkor a következő képet fogjuk látni:



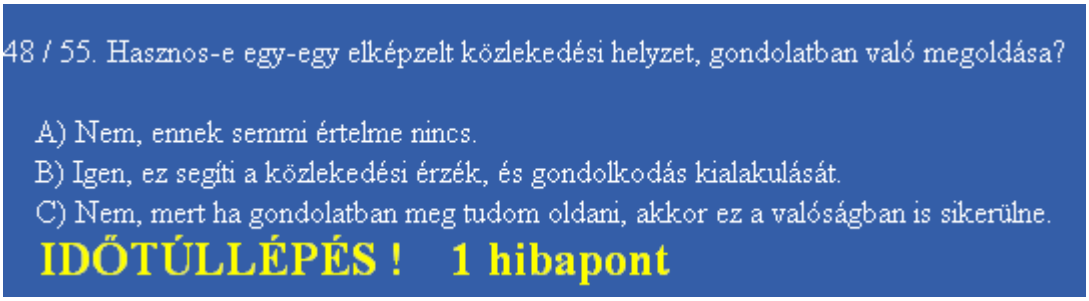
Ha ezt látjuk, akkor már biztosra vehetjük, hogy elfogyott az időnk. Ahogy láthatjuk is, a számítógép közli velünk, hogy a rendelkezésre álló idő elfogyott, nem engedi tovább oldani a feladatokat. Ami eddig elkészült azokat kijavítja, mert ugyebár az is könnyen előfordulhat, hogy az időtúllépés ellenére, sikeres vizsgát tettünk. Nézzük meg példának az első feladat javítását:



Láthatjuk, a kérdés után a válaszok alatt megjelent egy újabb információ, méghozzá az „Időráfordítás”. Ez tájékoztatja a felhasználót, hogy a kérdés megválaszolásával mennyi időt töltött el. Ebben az esetben a helyes válasz 6 másodperc alatt született meg.

Nagyon fontos! Ez a mező minden javításnál megjelenik, még akkor is, ha nem csúsztunk ki az időkorlátból, ha kihagytunk egy kérdést vagy akár akkor is, ha helytelen választ adunk a kérdésre.

Hogyan javítja tehát akkor a program, azokat a kérdéseket, amikre nem jutott idő. Ezt a következő ábra szemlélteti:



A program egész egyszerűen az időtúllépés üzenetet szúrja a feladat végéhez. Ezzel tudatja a felhasználóval, hogy erre a kérdésre nem maradt ideje. Ha a felhasználó csak 40 kérdést tudott megválaszolni, akkor a program a javításnál nem csak az elkészült 40 kérdést jeleníti meg kijavítva, hanem az összest. A maradék 15 feladatot a felhasználó persze nem is látta, de a program mindig 55 kérdést javít. Az „Időtúllépés” üzenet mellett a feladathoz tartozó hibapontot is láthatjuk. 3 pontos feladatra értelemszerűen 3 hibapont jár.

Az 55. kérdés után található „Vizsgabizottság döntése” linkre kattintva a program közli a sikert vagy a sikertelenséget.

B. Programozói szemmel

Az adatbázis

Alapok

Az program során egyetlen adatbázist használtam. A KRESZ nevű adatbázist. A programot alapvetően 3 főbb egységre lehet különíteni:

1. KRESZ
2. Vezetéselmélet
3. Műszaki ismeretek

A 3 egység szerkezetileg teljesen ugyanazokat a mezőket tartalmazza. Ezért tulajdonképpen szükségtelen 3 különböző táblát készíteni. Egyetlen táblára van szükség. A KRESZ, vezetéselmélet, és műszaki témakörhöz tartozó mezők ebben a sorrendben szekvenciálisan helyezkednek el. A KRESZ kérdésekből 1100 darab, a vezetéselméleti kérdésekből 188 darab és a műszaki ismeretek kérdéskörből pedig 53 darab kérdés található az adatbázisban.

Szerkezeti felépítése miatt, a tábla meglehetősen egyszerű. Az egyik mező felel azért, hogy meg lehessen a három kérdéskört különböztetni egymástól. A tábla egy sorának egyetlen kérdés felel meg, az összes információjával együtt. A kérdésen keresztül, a képeken át teljesen a válaszig, minden megtalálható a kérdéssel kapcsolatban.

A tábla létrehozása

1. *Otthoni környezetben (internet-kapcsolat nélkül)*

Először az otthoni környezetben való fejlesztéssel kezdeném, mivel így fejlesztettem a programot. Hiszen internet-kapcsolat nélkül is lehet PHP-ban programozni. Mellesleg sokkal gyorsabb is, mintha állandóan a módosított változatot töltenénk fel a világhálóra.

Legelső feladatunk, hogy nyissunk meg egy DOS-ablakot, és innen parancssorból a mysql.exe programot futtassuk. Ez elénk tár egy promptot: „mysql>” formában, ami jelzi számunkra, hogy a MySQL szerver várja az utasításainkat.

Készítettem először egy adatbázist, a CREATE DATABASE paranccsal. A MySQL mindenképpen megkövetel egy adatbázis, amiben a tábláinkat elhelyezzük. Itt tulajdonképpen elég volt egy adatbázist létrehozni, az egyszerű, stabil, és minél jobban átlátható program megalkotása érdekében. Ezek után csak annyi dolgunk van, hogy létrehozzuk a táblát. Ezt a CREATE TABLE adatbázis.tábla szintaxissal tehetjük meg.

Igenám, csakhogy ekkor van egy teljesen üres táblánk. Ezt fel kell tölteni, a fent említett három kérdéskörrel. A MySQL insert into parancsával kell itt dolgoznunk. De a probléma az, hogy több mint 1000 kérdés van. Az ugyanennyi utasítás. Mi lehet a megoldás?
Az, hogy írunk egy szöveges fájl, amiben az összes utasítás (kérdés) benne van. Mind a három kérdéskörnek egy-egy szöveges állományt hozunk létre. Innentől kezdve lényegesen lerövidült az egész művelet:

Source kresz;
Source vezelm;
Source muszaki;

Ezen három utasítás kiadásával a több mint 1000 utasításból álló munkafolyamat, pár másodperces időráfordítással kivitelezhető. Ha minden sikerült, akkor létrejön 3 darab fájl, táblanév.FRM, táblanév.MYD és táblanév.MYI nevekkel. Innentől kezdve lehet használni az adatbázist.

2. *Tábla létrehozása WEB-en*

Mivel az ingyenes Web-szerverek többsége támogatja a MySQL-t és a PHP programok futtatását, ezért nincs nehéz dolgunk. Itt viszont a parancssor helyett egy úgynevezett phpmyadmin nevezetű kis program segít feltölteni az adatbázisunkat a Web-szerverre.

Itt is ugyanúgy van lehetőség olyan fájlok megadására, amiben SQL utasítások sora van benne. Így nagyon hamar fel lehet tölteni az adatbázist. Egyetlen nagyon fontos dolog van, ami meg kell említeni, ez pedig nem más, mint a szerverhez való kapcsolódás szintaktikája. Míg a localhost-on keresztül való kapcsolódás így nézett ki:

```
<?php $sql = mysql_connect(localhost); ?>
```

Ami valljuk be, nagyon egyszerűen néz ki, addig a Web-szerverre a következő paranccsal tudunk csatlakozni:

```
<?php $sql = mysql_connect("SQL4.ULTRAWEB.HU", "nagymiki2", "*"); ?>
```

MySQL host: **SQL4.ULTRAWEB.HU**
Felhasználó név: **nagymiki2**
Jelszó: *

A tábla szerkezeti felépítése

Nagyon egyszerű táblát hoztam létre. Amely természetesen megfelel a célnak, és rendkívül áttekinthető. Úgyhogy nézzük is meg közelebbről!
A táblában a következő mezők helyezkednek el:

1. sorszam
2. tema
3. tipus
4. kerdes
5. valasz_a
6. valasz_b

.

.

.

30. valasz_z
31. megoldas
32. kep

1. *sorszam*

Ez az első mező, minden kérdésnek van egy sorszáma, 1. kérdés, 2. kérdés, stb. Ugyebár olyan nem lehet pl.: az 1. kérdés ugyanolyan sorszámmal benne legyen a táblában, mert nincs 2 db 1. kérdés. Mivel az egész táblában ez a mező teljesen egyedi és egyértelműen azonosít egy rekordot, ezért teljesen kézenfekvő, hogy nevezzük ki ELSŐDLEGES KULCSNAK.

2. *tema*

Ez pedig a kérdésekben használt témákat azonosítja. Rengeteg téma van, felfogható a teszt-könyv egy-egy fejezetének. Pl.: Egyenesen haladva, vagy vasúti pályával ellátott úttesten közlekedve, közúti baleset, a járművezetés emberi tényezői (ez utóbbi már vezetéselméleti téma). Ezzel tudjuk kiszűrni, hogy a felhasználó milyen jellegű kérdéseket akar gyakorolni.

3. *tipus*

A típus mező, a kérdéseknek a típusát tárolja el. Észre kell vennünk, hogy a téma és a típus mező nem ugyanaz. Rengeteg téma lehet ebben az adatbázisban, típusból viszont csak 7. Ezek a következők:

- könnyű
- táblás
- szituációs
- forgalmis
- alap
- elmélet
- műszaki

A könnyű kérdések igazából az első témakör kérdései a teszt-könyv első fejezetés öleli fel, ahol tulajdonképpen főként fogalmakat kér számon a felhasználótól. Ezt mindenkinek tudnia kell.

A táblás kérdések pedig csak és kizárólag az összes forgalmi jelzőtáblát tartalmazza, ami az úton fellelhető. Nem biztos, hogy csak a tábla nevét kell tudni, hanem néha azt is, hogy mit tilt, mit enged, és mit nem enged.

A szituációs kérdések olyan kérdések, amikkel a forgalomban közlekedve találkozhatunk. Pl.: keskeny úton kőomlás nehezíti a forgalmat, mikor 2 autó egyszerre ér oda. Kinek van elsőbbsége?

A forgalmis kérdések túlnyomó része nem más, mint egy kereszteződés, ahol egyszerre érnek oda a forgalom résztvevői. Itt főként a helyes továbbhaladási sorrendet kell megállapítani.

Az alap kérdések pedig olyanok, amiket mindenkinek tudnia kell. Ezek azok a kérdések, amit még moped kategóriára készülő felhasználóknak is tudni kell. A „B" kategóriásoknak is, és az összes többi ezek fölött elhelyezkedő kategóriának úgyszintén.

A következő csoport az elméleti típusú kérdések. A vezetéselmélet témakörét foglalja magába. Ez egy 188 darab kérdésből álló típus.

A műszaki kérdések: A műszaki alapismeretekből az összes olyan feladatot tartalmazza, amely a „B" kategóriára készülőknek kell.

4. *kerdes*

Ez egy olyan mező, amelyik a kérdés szövegét tartalmazza. Csupán egy string.

5. *valasz_?*

Ezek a válasz mezők. Amik tartalmazzák a válaszokat. 26 darab válaszmezőt tartalmaz minden egyes rekord. Igaz, lehetne mondani, hogy kicsit pazarlóan bánunk a merevlemez kapacitásával, de ugyanakkor sokkal egyszerűbb kezelni az adatbázist, úgymond fix hosszúságú mezőink vannak, a változó hosszúság mellett. Az előbb említett kérdésre adja meg a válaszlehetőségeket. Lehet, hogy csak 3 db válasz van.

Ekkor természetesen a többi mező a NULL értéket kapja meg, ami jelzi, hogy nincs olyan válasz. Pl.: valasz_a, valasz_b, valasz_c létezik, azaz rendesen értéket kap, de már valasz_d értéke NULL lesz.

6. *megoldas*

Ez a mező a megoldásnak a betűjelét tartalmazza. Pl.: 'a' vagy 'b' vagy bármi más. Ezzel jelzi, hogy a kérdés mezőben megjelenő kérdésre, a valasz_? Mezőkben megjelenő válaszok közül melyik betűjelű a helyes megoldás. Csak egyetlen jó válasz lehetséges. Nincs megengedve a több jó válasz lehetősége, mint az amerikai típusú teszteknél.

7. *kep*

Ez a mező annak a képfájlnak tartalmazza a nevét, mely a kérdéshez tartozik. Gondoljunk csak bele, hogy tényleg vannak olyan kérdések, amelyekhez kép is tartozik. Pl.: a forgalmi helyzetes, a szituációs kérdéseknél fordul elő nagyobb számban. Mert ha egy olyan kérdéssel állunk szemben ahol a kérdés:
Továbbhaladhat-e ebben a forgalmi helyzetben? Válaszok: Igen, Nem.
Ha nem látjuk a képet, akkor esélytelen az egész feladatot megoldani. Ezért a képeket egy külön könyvtárban tároljuk, ahonnan a „kép" mező alapján csak a fájlnevet kiolvassuk, és már meg is tudjuk jeleníteni a feladathoz tartozó képet.

A mezők típusai

- A sorszam mező típusa INTEGER. Csak sorszámozott egész típust kaphat, ez az elsődleges kulcs. Értéke sohasem lehet NULL.

- A tema mező, mivel szöveges információt tartalmaz nem más mint egy string. VARCHAR(255) típust kapott. Értéke sohasem lehet NULL.

- A tipus mező szintén VARCHAR(255) típusú mivel ez is string. Ez sem lehet NULL értékű.

- A kerdes mezőre is ugyanazok vonatkoznak, mint az ezt megelőző két mezőre.

- A valasz_? mezőkkel szintén ugyanez a helyzet, annyi különbséggel, hogy itt először jelennek meg az úgynevezett NULL értékű mezők, melyek azt jelzik, hogy az adott mezőnek nincs értéke. De ami nagyon fontos, hogy az összes mező nem lehet NULL, mert egy megoldásnak mindenképpen léteznie kell.

- A megoldas mező: szöveges típusú mező, viszont itt annyi a helyzet a több mezővel ellentétben, hogy itt a mező 1 karakter hosszú. A rend kedvéért ennek a mezőnek is 255 hosszúságot engedek meg, attól függetlenül, hogy ezt a hosszt sohasem éri el.
  Itt nem lehetnek NULL értékű mezők.

- A kep mező: mint azt fentebb is írtam, ez egy fájlnevet tartalmaz, ami nem lehet más csak szöveg. Ezért ez is VARCHAR(255) típust kapott. NULL értékű mező meg van engedve.

Összefoglalva azt lehet megállapítani, e mezők tükrében, hogy az adatbázisban kétféle típust használok: Az egész típust és a string típust. Melyek közül egyes mezők felvehetik a NULL értéket, másik mezők viszont nem.

A programkód rövid áttekintése

A program 5 php fájlból áll. Ezek a következők:

- index.html
- menu.php
- menu1.php
- vizsgakezd.php
- vizsga.php

index.html

A Web-szerveren léteznie kell egy index.html nevű fájlnak. Mikor el akarjuk érni a programot és beírjuk a: www.nagymiki2.uw.hu címet akkor a Web-szerver ezt az index.html fájlt kezdi el keresni, és ezt is indítja el először. Ez az úgynevezett kezdő vagy bejelentkező képernyő.

Nagyon fontos szerepe van ennek a résznek. De előbb beszéljünk az úgynevezett munkamenet változókról. Ez egy olyan változó, amely egy munkafolyamatot elindít, vagy leállít. Mi a lényege tehát? Én egy nagyon fontos tulajdonságát használtam ki: a paraméterátadást.
A program egy Web-oldalt többször is meghív, lépked weboldalak között. A munkamenet változókkal a változók, a paraméterek nem vesznek el, hanem megmaradnak.
Az úgynevezett $_SESSION[’változó’] tömbökről van szó.

A PHP munkamenet kezelése lehetővé teszi adatok megőrzését az egymást követő oldal lekérdezések között. Ez képessé tesz még testreszabhatóbb oldalak készítésére.

Ha valaki erre a weboldalra látogat, akkor egy egyedi azonosítót kap, az úgynevezett munkamenet azonosítót (session azonosítót). Ez vagy egy sütiben (cookie) tárolódik a látogató gépén, vagy az URL-ben közlekedik oldalról oldalra.

A munkamenet támogatás lehetővé teszi tetszőleges számú változó megőrzését a PHP oldal lekérdezések között. Ha egy látogató érkezik webhelyre, akkor vagy elkezdődik, vagy folytatódhat a munkamenete. PHP egy munkamenet azonosító érkezését várja. Ha nem érkezik, új munkamenetet indít.

Manuálisan a **session_start()** függvénnyel, tudunk munkamenetet folytatni / megkezdeni. Ha egy érvényes azonosító érkezett, a korábban beállított munkamenet környezet visszaállításra kerül.

Minden a munkamenethez rendelt változó szerializálódik a PHP oldal futásának befejeződésekor. A nem definiált, de munkamenethez rendelt változók a későbbi folytatásokban nem jönnek újból létre.

A $_SESSION változót szokták még szuper-globális változónak is nevezni. Ez a változó a php 4.1.0 verziójától kezdve a rendelkezésünkre áll, csakúgy mint a $_POST, $_GET változók.

Aki a programot használja, láthatja, hogy a program többször is visszalinkel minket a főmenühöz. Ha hiba lép fel, vagy akár akkor is, ha befejeztük a vizsgát, de még sok számos esetben is. Ekkor ugye már újból indul a program és nincs szükség semmiféle változóra. Tömböket üríteni kell és változókat felszabadítani.

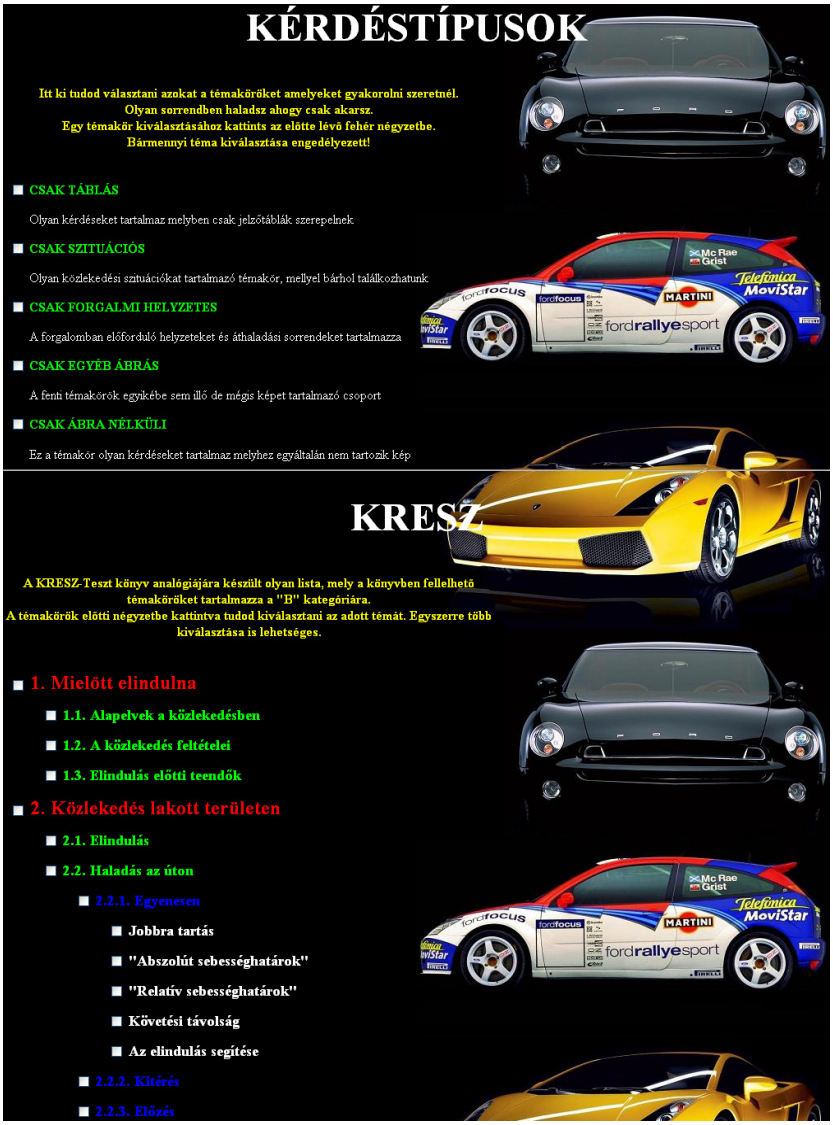
Viszont azt nem tudjuk, hogy aki oda került az most jár a Web-helyen először vagy már a program használata közben került a főmenühöz. Ezért két utasítást adok ki a program elején:

```
<?php  session_start(); session_unset(); ?>
```

A session_start() egy új munkamenete indít el vagy folytat. A session_unset() pedig felszabadítja az eddig használt munkamenet változóinkat. Úgynevezett „Tiszta lappal” indulhatunk az alkalmazásnak.

menu.php

Ennek a résznek az a feladata, hogy előállítson egy formot. Egy olyan formot melyben minden elem előtt egy jelölőnégyzet szerepel. Ezt látja a felhasználó:



Ennek végső soron semmi egyéb szerepe nincs az adatgyűjtésen kívül. Felméri, hogy a felhasználónak mire van igénye. Milyen kérdéseket akar gyakorolni. A felhasználó kipipálja a kívánt kérdéstípust és/vagy a kívánt feladatkört, azaz fejezetet.
Itt látszik, hogy más-más címet kapnak a fejezetek. A főtémakörök pirosat, az altémák pedig zöldet, azon belül kék majd végül a fehér szín jelzi a legszűkebb témaköröket.

Ha kész, akkor az OK gomb megnyomásával a felhasználót a program átlinkeli a másik programra, aminek a neve: menu1.php
Van olyan eset is amikor a felhasználó véletlenül kattintott erre a gyakorló részre vagy meggondolja magát. Erre az esetre tettem bele a programba egy visszamutató linket, amivel a felhasználó visszajuthat a főmenühöz. „Vissza a főmenühöz”

menu1.php

Ez a program a gyakorló rész magja. Azt, hogy milyen elemeket milyen típusú kérdések kerültek kiválasztásra azt az előző form $_POST változója tartalmazza, ami egy asszociatív tömb. Miután már tudjuk, hogy mit választott ki a felhasználó, ezek után annyi dolgunk van, hogy azokat a típusú kérdéseket ki kell választani az adatbázisból. Bár ez a rész ettől azért egy kicsivel összetettebb. No lássuk csak:

1. A select utasítás megalkotása a nyert információkból

   Létrehoztam egy tömböt mely annyi elemű, ahány jelölőnégyzet van. Minden jelölőnégyzethez tartozik egy select utasítás a tömbből. Pl.: a nyolcadik jelölőnégyzethez a tömb nyolcadik elemét rendelem hozzá, ami például lehet olyan, hogy : tema='A közlekedés feltételei';
   Nekem már csak annyi dolgom van, hogy a nyert információk alapján a tömbből kiolvassam a select parancsot. Több jelölőnégyzet esetén OR paranccsal kapcsolom össze a tömbben található parancsokat Ha ez kész akkor a kérdéstípusokat (csak táblás, csak szituációs, csak forgalmi helyzetes, stb.) a már meglévő OR kapcsolattal előállított stringet pedig AND kapcsolattal illesztem a string végére. Pl.: tema='A közlekedés feltételei' AND tipus='csak táblás';

2. adatbázis lekérdezése

   Ha kész van a select stringünk, akkor nincs más dolgunk, mint hogy lekérdezni az adatbázisból a kellő rekordokat egy select utasítással. Először viszont csatlakoznunk kell az adatbázishoz. Ha bármilyen jellegű probléma merülne fel, akkor a program jelzi azt:
   - "Nem sikerült kiválasztanom az adatbázist" ezt főleg akkor, ha nem létezik az adatbázis vagy a nevét írtuk rosszul.
   - "HIBA TÖRTÉNT AZ ADATBÁZIS LEKÉRDEZÉSE SORÁN!!!" ezt pedig akkor, amikor már túl vagyunk a kapcsolódáson, csak a lekérdező utasítás,

azaz a select-ben vannak hibák, vagy teljesen üres az adatbázis, illetve lehetnek még szintaktikai és szemantikai hibaforrások is.

Ha nincs probléma, akkor egy változóban visszakapjuk az eredménysorokat, és arról is információt kapok, hogy hány sora lett az eredménynek.

3. kérdések megjelenítése sorrendben

   Miután lekérdeztük az adatbázist, annyi dolgunk van, hogy a kérdéseket a megfelelő formában a felhasználó elé terjesszük. A program információkat ír a jelenlegi helyzetről. Hány rossz, hány jó válasz, hol járunk a kérdéssorozatban, stb.
   Egy kérdést 2x fog látni a felhasználó. Először amikor választania kell a lehetséges válaszok közül, másodszor pedig mikor a számítógép kijavítva, megmutatja neki, hogy jól válaszolt-e vagy sem. Tehát megmutatja a helyes választ.
   Ezek után jön a következő kérdés és a hozzá tartozó javítás. Mindaddig, amíg a kérdések el nem fogynak.

4. Végeredmény

   A kérdéssorozat legvégén a program összesít. Megmutatja hogy összesen hány jó illetve hány rossz válasza volt a felhasználónak. Természetesen kiírja százalékos formában is, hogy tisztán lássuk, vajon mennyit kell még fejlődnünk ebben a témában.
   Található még itt egy „Vissza a gyakorláshoz” link, ami visszavezet minket a gyakorláshoz, ahol kezdhetünk újabb gyakorlást.

Nagyon fontos! Mivel nem a főmenühöz kerülünk vissza, hanem a gyakorló részhez, ezért ki kell adni a program elején a session_unset(); utasítást a munkamenet változók felszabadításához. Itt is teljesen üresnek kell lennie a $_SESSION tömbnek.

vizsgakezd.php

Mivel a vizsga.php-ben (következő téma) már nem szabad törölni a session változókat, ezért közbeiktattam egy lapot, melyen ezt megteszem. Mivel mindig ugyanaz a lap hívódik, meg ezért nem lehet azt, hogy minden egyes hívásnál ürítem a session változókat.

A következő két parancs van ebben a részben:

1. session_start();
2. session_unset();

Van egy másik fontos szerepe ennek a résznek. Gondoljunk csak bele, hogy ha megoldottunk egy teljes 55 kérdéses vizsgát, akkor a munkamenet változók a memóriában maradnak. Ha valaki el akar kezdeni egy újabb vizsgát, akkor a munkamenetet megint inicializálni kell, azaz minden munkamenet változót fel kell szabadítani a memóriából.

vizsga.php

Ha ezt a részt választjuk, akkor már nem a gyakorlás, hanem egy éles vizsga vár minket. 55 kérdésből álló tesztlapot kell generálni, bizonyos fajta kérdésekből. Összesen hétféle kérdésből kell a vizsgázó tudását ellenőrizni.

1. adatbázisból való tömbfeltöltés

A hét típus a következő: könnyű, táblás, szituációs, forgalmis, alap, elmélet, műszaki.
Külön-külön minden típust le kell kérdezni az adatbázisból. És a kapott eredményből kiválasztani a kellő számú feladatot.
Kezdjük a könnyű típussal. Lekérdezem az összes könnyű típusú kérdést kapott pl.: 110 eredményt. Ebből kell kiválasztani véletlenszerűen 10-et.

- Könnyű kérdésből 10-et
- Táblás kérdésből 10-et

- Szituációsból 5-öt
- Forgalmis kérdésekből 5-öt
- Alap kérdésekből 10-et
- Vezetéselméletből 10-et
- Műszaki kérdésekből pedig 5-öt

Így jön ki az 55 kérdéses tesztsor, amit valós életben is használnak. Nagyon fontos figyelni azt, hogy ne legyen két egyforma kérdés, mivel véletlenszerűen választ a program. Ezt mindig ellenőrizni kell a forráskódon belül.
Az adatbázisból kiolvasva a következő mezőkre lesz szükségünk: sorszám és típus.
Viszont tárolni kell azt, hogy az adott kérdésekre hány pont jár. Illetve hányadik oldalon található a kérdés. Ezt egy egyszerű tömbbel meg lehet oldani.

Ha egy típusból kiválasztottuk a megfelelő mennyiségű kérdést, akkor azt sorszám szerint rendezem, és egy új tömbbe, egy úgynevezett gyűjtőtömbbe gyűjtöm a feladatokat. Mindig hozzáfűzöm a tömb végéhez. A végére pedig benne lesz az 55 kérdés és kategóriánként sorszám szerint rendezve.

2. A feladatok képernyőre írása

Ha kész a tömbünk, akkor a feladatokat egymás után fel kell tenni a felhasználónak. Egy feladatot felteszünk és várjuk rá a választ. Hogy ne hozzuk ki a gondolatmenetből, és ne zavarjuk a felhasználó gondolkodását, ezért nem írunk ki semmi információt, azt illetően, hogy jó volt-e illetve rossz volt-e a kérdésre adott válasza. A háttérben viszont a kérdéseire adott válaszokat el kell tárolni, hiszen csak ezen információk tudatában tudja majd a számítógép kijavítani a vizsgát. Egy tömbben tároljuk a válaszokat.

3. Feladatok kijavítása

Miután végighaladt a felhasználó mind az 55 kérdésen, akkor a számítógép kiírja, hogy vége a vizsgának, és ismét megmutatja a feladatokat kijavítva. Ez nagyon fontos a megfelelő és hatékony tanulás szempontjából, hiszen a felhasználó tud tanulni a saját hibáiból. Bár még hatékonyabb, ha le is írja egy papírra, hogy miket rontott el.

4. „A döntés”

Miután a felhasználó látta és megértette, hogy miket rontott el egy következő oldalra kalauzolja el minket a program. Mégpedig oda ahol tisztán le van írva, hogy a felhasználó átment-e a KRESZ vizsgán, avagy sem. Ezek után egy linkre való kattintással visszakerülünk a főmenühöz. A felhasználó ennek tükrében már tudni fogja, hogy oldjon még egy vizsgát, vagy inkább fektessen nagyobb hangsúlyt a gyakorlásra.

IV. Összefoglalás

Végére értünk a program bemutatásának. Láthattuk, miként épül fel az alkalmazás. Belekóstolhattunk a programozás ezen területére. Lehet, valaki kedvet kap majd ezek után ennek a páratlan és nagyon kényelmes eszközrendszer használatához. Az is lehet, hogy valaki inkább mást választana. Véleményem szerint azt meg kell említeni, hogy vagy használjuk ezt a technikát vagy nem sose szabad azt az egyet elfelejteni, a WEBEN történő alkalmazásfejlesztés egyre jobban teret hódít magának. Nap mint nap, a számítógép elé ülünk, néha rá vagyunk kényszerülve, hogy használjuk, de ki nem lehet kerülni. Ha pedig a világhálón szörfözünk, akkor is rengeteg olyan webhellyel találkozunk, melynek hátterében egy adatbázis-kezelő tevékenykedik.

Ezt az alkalmazást bárkinek tudom ajánlani, aki „B" típusú gépjárművezetői jogosítvány megszerzésére készül. Az előadás anyaga melyet az erre megfelelő tanfolyamokon sajátíthatunk el, természetesen elengedhetetlen a kellő siker érdekben. A programnak viszont egyik nagy előnye a Teszt-könyvvel szemben az, hogy a tanulásra szánt időt lényegesen lerövidíti. Nem a tanulónak kell egy-egy tesztlap után még azzal is bajlódnia, hogy a helyes válaszokat egyeztesse a könyvben, hiszen a program rögtön javít. A könyvben nincs arra lehetőség, hogy különböző témájú és típusú feladatokat kombináljunk össze igényünk szerint. A programmal tehát célirányosan tanít. A karbantartásról csak annyit, hogy a program mindig újabbnál újabb ötletekkel bővül, és követi a KRESZ változásait. Mindent összegezve, véleményem szerint a program minden esetben csak javára válhat a tanulónak.

Személyes tapasztalatom a programmal kapcsolatban annyi, hogy én is ezt az alkalmazást használtam Teszt-könyv helyett mikor 2006 nyarán a KRESZ vizsgámra készültem. Nem is gondoltam volna, de a siker óriási volt. Egy hibám sem volt a vizsgafeladatokban. Rajtam segített a program, és remélem másokon is fog.

Én, mint programozó megláttam a PHP+MySQL+Apache eszközhármas szakmai jelentőségét, mivel az adatbázis-kezelésre a valós világ bármely területén szükség lehet. Teljesen mindegy hogy csak egy DVD kölcsönző nyilvántartását, vagy akár egy kórházi beteglistát akarunk kezelni, adatbázisra mindkét esetben szükség van. Az emberek nem

vihetik magukkal a számítógépüket - jóllehet ma már a laptop számítógépek kezdenek egyre jobban elterjedni – és mivel a világháló már szinte a föld szinte bármely háztartását bekebelezi, ezért az az egyik legkényelmesebb dolog, hogy bárhova megyünk, legyünk bárhol, a megszokott programjaink mindig kéznél vannak. Futnak minden számítógépen, melyek csatlakozatva vannak az internetes világhoz. Nekünk csak egyetlen dolgot kell tenni: egyet kattintani! Hisz ahogy mondani is szokták: „A világ csak egy kattintásra van!”.

V. Irodalomjegyzék

- *Feladatok, szövegek, képek, megoldások*
    a. A járművezetői vizsga TESZT kérdéseinek gyűjteménye 8. átdolgozott kiadás (I. rész – Közlekedési ismeretek II. rész – Járművezetés elmélete III. rész – Szerkezeti és üzemeltetési ismeretek)

- *Php-ről*
    a. http://hu.php.net/intro-whatcando
    b. http://www.pergel.hu/phpdoksi/intro-history.shtml
    c. http://www.prog.hu/cikkek/881/A+kiszolgalo+kornyezet+inditasa.html

- *MySQL-ről*
    a. http://hu.wikipedia.org/wiki/MySQL
    b. http://www.prog.hu/cikkek/3/A+MySQL.html

- *Apache-ról*
    a. http://www.prog.hu/cikkek/421/Fejlesztoi+koryezet+kialakitasa+1.html

VI. Köszönetnyilvánítás

Köszönöm hasznos tanácsait, a dolgozat elkészítésében nyújtott segítségét és szakmai útmutatásait témavezetőmnek

Dr. Rutkovszky Edéné *egyetemi tanársegédnek.*